GESIS

IZ InformationsZentrum Sozialwissenschaften

IZ-Arbeitsbericht Nr. 1

Das WOB-Modell

Jürgen Krause

Dezember 1995



Lennéstraße 30
D-53113 Bonn
Tel.: 0228/2281-0
Fax.: 0228/2281-120
email: krause@bonn.iz-soz.de
Internet: http://www.social-science-gesis.de

ISSN:	1431-6943
Herausgeber:	Informationszentrum Sozialwissenschaften der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V. (ASI)
Druck u. Vertrieb:	Informationszentrum Sozialwissenschaften, Bonn Printed in Germany

Das IZ ist Mitglied der Gesellschaft Sozialwissenschaftlicher Infrastruktureinrichtungen e.V. (GESIS), einer Einrichtung der Wissenschaftsgemeinschaft Blaue Liste (WBL)

Inhalt

1 Einleitung

Beim WOB-Modell (auf der Werkzeugmetapher basierende strikt objektorientierte graphisch-direktmanipulative Benutzungsoberfläche) handelt es sich um ein aufeinander abgestimmtes Bündel softwareergonomischer Vorschläge, die in ihrer Gesamtheit zu einer effizienten und „natürlichen“ Gestaltung von Benutzungsoberflächen (BOF) für Informationssysteme führen sollen. Idealiter schließt das WOB-Modell eine Lücke bei der Softwareentwicklung: Entwickler von BOF, die Vorschriften in den Styleguides zu Rate ziehen, finden dort zwar klare „Man nehme ...“ Regeln nach dem Muster von Kochrezepten, merken aber bald, daß deren Widersprüchlichkeit im Einzelfall im Wesen der Softwareergonomie liegt. Orientieren sie sich andererseits an allgemeinen kognitionspsychologischen Erkenntnissen und Theorien bzw. Globalanforderungen wie der nach Transparenz, läßt sich kein Operationalisierungsverfahren angeben, das konkrete Gestaltungsprobleme lösen könnte.

Softwareergonomie, wie sie heute bei der Realisierung graphischer Oberflächen betrieben wird, ist zum Großteil nur auf der Basis kreativer Gestaltungsideen vor dem Hintergrund eines breiten softwareergonomischen Fachwissens effizient möglich. Styleguides vermeiden - für sich genommen - nur die gröbsten Fehler. Da Softwareentwicklung heute einen Massenmarkt bedient, führt diese Ausgangssituation zum unbefriedigenden Zustand der meisten kommerziell verfügbaren BOF.

Letztlich geht es bei der Formulierung des WOB-Modells um die Zusammenfassung und Abstraktion softwareergonomischer Erfahrungen, die seit Ende der achtziger Jahre am FG Informationswissenschaft der Universität Regensburg und seit Mai 1995 am IZ Sozialwissenschaften in Bonn (GESIS) gemacht wurden.

Ausgangspunkt für das WOB-Modell waren empirische Tests im Projekt WING-IIR, bei dem es um die Entwicklung eines multimodalen Werkstoffinformationssystems ging (cf. Krause et al. 1993/94 als Überblick). Gefördert wurde das Projekt 1989 - 1995 vom Bundesministerium für Wirtschaft (Förderkennzeichen Wi 712 50). Industrielle Partner waren die MTU in München und das europäische Werkstoffinstitut in Petten.

Seit Mitte 1993 wurde versucht, die am Werkstoffsystem der MTU gemachten Erfahrungen auf andere Anwendungen zu übertragen, zuerst im Werkstoffbereich selbst (Hochtemperaturdatenbank HTM des EU Institute for Advanced Materials in Petten und IMA - Werkstoffdatenbanken in Dresden), danach auf andere Gebiete. Das WOB-Modell diente als Basis für Designvor-

schläge von Versicherungssoftware (NKK Regensburg), für die Formularkomponente der Bürosoftware COMFOWARE, (SNI Augsburg/München), für das Beratungsprogramm einer Bausparkasse, für die Textdatenbanken der DATEV und derzeit für die Literatur- und Forschungsdokumentationen des IZ Sozialwissenschaften in Bonn.

Bisher entstanden drei kommerzielle Softwareentwicklungen, denen das WOB-Modell zugrunde liegt:

- der Private Organizer Version 3 der Mapware Datensysteme (Regensburg),
- ein Produkt der Deutschen Postreklame in Frankfurt, bei der es um die Abfrage der weißen und gelben Seiten geht
- und das Werkstoffinformationssystem der MTU (München).

Seit Januar 1995 wird am IZ Sozialwissenschaften in Bonn (GESIS) in einem Forschungsverbund mit dem Zentralverband der Elektroindustrie (ZVEI, Frankfurt) und den Wirtschaftsinstituten DIW und IFO (Berlin, München) auf der Basis des WOB-Modells das Verbandsinformationssystem ELVIRA entwickelt, das seit Januar 1996 in einem ersten Prototyp vorliegt (Förderung durch das Bundesministerium für Wirtschaft (Förderkennzeichen: II C7-003060/10); cf. Krause/Mandl/Stempfhuber 1996).

In all diesen verschiedenen Anwendungsbereichen ließ sich das WOB-Modell anwenden, so daß davon ausgegangen werden kann, daß eine Generalisierung des Ansatzes erfolgversprechend ist.

Die erste Formulierung der WOB-Prinzipien erfolgte in Krause 1994. Der vorliegende Text berücksichtigt die seitdem gemachten Erfahrungen und Veränderungen. Ausweitungen des WOB-Ansatzes finden sich darüber hinaus in drei Dissertationen zum WING-Projekt (Wolff 1996, Marx 1995, Roppel 1996), die sich mit Visualisierungskomponenten und der natürlichsprachlichen Zustandsanzeige befassen.

Eine Zusammenfassung der Ergebnisse des WING-Projektes enthält Krause/Womser-Hacker 1996.

2 Generelle Probleme bei der Gestaltung graphischer Oberflächen

Als Grundparadigma „natürlicher“ Oberflächen hat sich spätestens seit Ende der achtziger Jahre der graphisch-direktmanipulative Modus durchgesetzt (cf. Word für Windows Version 6 und die Windows95 Version als typische Beispiele). Will man Software im Rahmen dieses Paradigmas entwickeln, scheint es erst einmal keine Probleme zu geben.

- Man zieht eine Reihe von Styleguides sowohl von den Betriebssystemherstellern (z. B. den SAA Styleguide von IBM) als auch von Firmen (z. B. den Siemens Styleguide 1992) heran.

  Hier bekommt der Softwareentwickler - nach einem allgemeinen Teil - wie in einem Rezeptbuch klare Detailanweisung, z. B. wie viele Einträge ein Menü haben soll und welche Gestaltungselemente wann Sinn machen.

- Es gibt rechtsverbindliche Normen, die die Gestaltung im Sinne einer von Arbeitnehmern einklagbaren Forderung regeln (ISO 9241, DIN 66234).

- Darüber hinaus findet man von der Arbeitswissenschaft bis zur Kognitionswissenschaft eine Reihe kognitiver (Teil)theorien zur Perzeption und menschlichen Informationsverarbeitung (cf. Pinker 1990 und Wolff 1996: Kap. 5 als Überblick).

Warum gibt es dann trotzdem so viele softwareergonomisch schlecht realisierte Produkte? Daß das so ist - trotz aller gegensätzlicher Beteuerung und trotz eines Paradigmas, das die „natürliche“ Bedienung zum Kern der Designbemühungen macht, zeigt der wachsende Unmut der Benutzer. Hier nur ein Beispiel aus der Computerpresse:

> „Wo rohe Kräfte sinnlos walten... Es scheint ein Naturgesetz zu sein: Wann immer sich die Softwarehäuser daran machen, dem Anwender das Leben zu erleichtern, beschwören Sie Usability-Katastrophen herauf. Ein Gruselkabinett ...Da hat man häufig das Gefühl, Frankenstein hätte bei der Programmentwicklung mitgemischt...Dennoch erinnern die Usability-Versuche der Softwareentwickler an die Draisinen des 19. Jahrhunderts, die schwerfälligen, kuriosen Hochräder des Freiherrn von Drais ...“ (WIN, Februar 1993, S. 36ff).

2.1 Kenntnisstand und Anwendung softwareergonomischer Regeln

Eine wichtige Ursache schlechter softwareergonomischer Gestaltung liegt darin, daß softwareergonomisches Wissen bei den kommerziellen (und universitären) Entwicklungsgruppen heute noch kaum vorhanden ist. So kamen Beimel/Hüttner/Wandke 1992 bei einer Befragung von 82 Programmierern in 50 Softwarefirmen zu folgendem Ergebnis:

- Nur 22% kannten die DIN-Norm,
- nur 1% die ISO-Norm.
- 2/3 der Softwareentwickler verfügten über „keinerlei" softwareergonomische Erfahrung.

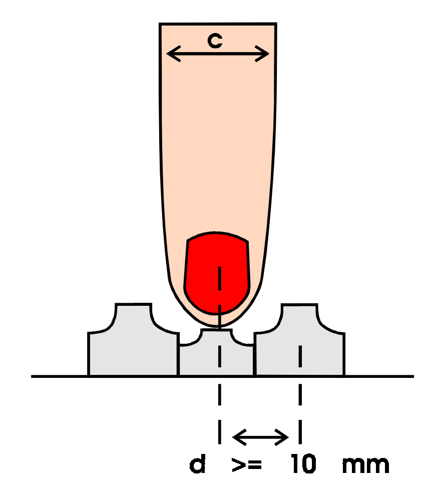
Die zweite Ursache spricht eine Voraussetzung an, auf die die DIN- und ISO-Normen ausdrücklich hinweisen: Die Anwendung der Softwareergonomieregeln setzt eine Aufgabenanalyse und Benutzerpartizipation voraus, zwei Forderungen, die fachlich heute nicht mehr umstritten sind.

Wenn in der Praxis oft darauf verzichtet wird, liegt das auch daran, daß die Zielgruppe z. B. von Textverarbeitungssystemen aus Marktüberlegungen heraus so weit gefaßt werden soll, daß sich keine ergonomisch homogenen Gruppen mehr ergeben.

2.2 Inhärente Widersprüchlichkeit softwareergonomischer Regeln und Entwicklungsdynamik

Auch wenn die „Selbstverständlichkeiten" aus Abschnitt 2.1 beachtet werden, ist man vor allem aus zwei Gründen noch weit von einer befriedigenden Gestaltung entfernt:

a) Es gehört zum Wesen ergonomischer Regeln, Widersprüche zu erzeugen. Am besten läßt sich dieses Phänomen an einem Beispiel aus der Hardwareergonomie klar machen (cf. Koch/Noé 1980:55):



Beispiel Schreibmaschinentastatur aus Koch/Noé 1980:55

Empirische Tests zeigen, daß die Schreibgeschwindigkeit um so höher wird, je enger die einzelnen Tasten zusammenrücken. Das Phänomen läßt sich auch an Laptops mit kleineren Tastenfeldern beobachten, bei denen Benutzer eine höhere Schreibgeschwindigkeit erreichen als an den „komfortablen" Büro-PCs. Leider gibt es zusätzlich eine gegenläufige Gesetzmäßigkeit, die sich ebenfalls empirisch belegen läßt: Je enger die Tasten zusammenrücken, um so mehr Schreibfehler stellen sich ein, da die Finger öfter die Nachbartaste mitnehmen.

Die Widersprüchlichkeit der ergonomischen Regeln ergibt sich aus der Gegenläufigkeit bei der Optimierung zweier gewünschter Eigenschaften: Optimale Eingabeschnelligkeit und weitestgehende Fehlerfreiheit schließen sich bei der Operationalisierung aus.

Die „klassische" Lösung dieses ergonomischen Widerspruchs besteht darin, eine für beide Anforderungen suboptimale Lösung zu finden, bei der man - bezogen auf beide Anforderungen - die wenigsten Abstriche machen muß. Man rückt die Tasten deshalb so weit zusammen, daß der Benutzer „nicht zu viele" Fehler macht, aber noch „erträglich" schnell schreiben kann.

Das obige Bild zeigt jedoch, daß ein besserer Weg aus dem Dilemma der inhärenten Widersprüchlichkeit ergonomischer Regeln gefunden wurde: Es wird eine Eigenschaft des menschlichen Fingers ausgenutzt: er ist vorne abgerundet. Mit dem Wissen um diese Eigenschaft und einer

„kreativen Idee“ läßt sich durch spezifisch gestaltete Tasten ein wesentlich besserer Ausgleich beider Anforderungen finden.

Dieses am einfachen Beispiel der Tastaturgestaltung verdeutlichte Schema: (zumindest teilweises) Auflösen der Widersprüchlichkeit durch die Verbindung spezifisch menschlicher Eigenschaften mit einer kreativen Grundidee gilt nicht nur für die Hardware-, sondern gleichermaßen für die Softwareergonomie.

Ein einfaches Beispiel gegenläufiger Regeln aus der ISO-Norm ist die nach Adaptionsmöglichkeiten (z. B. Menüanpassung) und die nach Konsistenz (cf. Fox 1993:77).

b) Ein zweiter Grund für die Schwierigkeiten der Softwareentwickler ist die Entwicklungsdynamik des Feldes, die immer wieder zu ungeregelten Bereichen führt. Die Industrie produziert fast täglich neue Gestaltungselemente, für die es noch keine wissenschaftlich fundierten Überlegungen, geschweige denn empirische Erfahrungen gibt. Beispiele der letzten Zeit sind das „Notebook“ von OS\2 und die entsprechenden Registerkarten von MICROSOFT.

Ein weiterer Beispielbereich sind die multimedialen bzw. -modalen Schnittstellen, mit der die Verbindung von Graphik mit Text, Sprache, Bildern, Animationen und Videosequenzen angesprochen wird (cf. Fox 1993:78). Ihre Gestaltung regeln die ISO-Normen kaum. Dies ist auch kein Wunder, da es in der Forschung erst wenige Ansätze einer Mischung von Modalitäten gibt (cf. Maybury 1993 als Überblick). Sie gehen in der Regel über trivialerweise plausible Verbindungen wie die Integration von Bildern in ein Malereilexikon, von Videosequenzen bei Reiseinformationssystemen u.ä. nicht hinaus. Auf der Seite Abfragekomponenten befaßt man sich vor allem mit der Integration von Zeigegesten in natürlichsprachliche Abfragen (cf. aber Krause 1993a).

Wann welche Modalität auf der Abfrageseite vorzuziehen ist und wie sie sich sinnvollerweise koppeln lassen, bleibt zu klären.

Bei all den bestehenden Unsicherheiten wäre es somit eine große Hilfe, wenn sich ein relativ geschlossenes - aber nicht abgeschlossenes - softwareergonomisches Modell entwickeln ließe, daß die Frage der Modalitätsmischung zumindest für die Bereiche Graphik und natürliche Sprache

regelt und die potentiellen Widersprüche softwarergonomischer Regeln auflöst.

2.3 Methoden der Softwareentwicklung

Weniger Unsicherheit besteht dagegen bei den Methoden, die für die softwareergonomische Entwicklung auszuwählen sind. Man ist sich in der wissenschaftlichen Literatur relativ einig, daß extensive Aufgabenanalysen, rapid prototyping und empirische Tests mit Benutzerpartizipation die Basis jeder BOF-Entwicklung bilden sollten (cf. z. B. Whiteside et al. 1988).

Dies sind auch die Methoden, die im Projekt WING-IIR angewandt wurden (cf. Krause et al. 1993/94).

3 Ausgangssituation: Vorgehensweise und Ansatzpunkte im Projekt WING-IIR

Vor der Diskussion einiger Lösungen des WOB-Modells für die Probleme in den Abschnitten 2.1 und 2.2 soll kurz die Ausgangssituation im Projekt WING-IIR geschildert und die empirische Basis verdeutlicht werden. Dies erscheint wichtig, da das Modell im gegenwärtigen Zeitpunkt seine empirische Legitimation vor allem aus den umfangreichen WING-Tests ableitet. Die WING-Tests umfaßten bei der ersten Formulierung des Modells 1994 etwa 100 Stunden mit der Videokamera aufgezeichnetes und ausgewertetes Material (cf. Krause/Womser-Hacker 1996).

3.1 Vorgehensweise

Schon nach einer ersten Aufgabenanalyse bei der MTU war klar: Keines der bisher gängigen Retrievalmodelle wie hierarchischer Zugang, Hypertext oder natürlichsprachliche Anfrageformulierung erschien ausreichend, um die Bedürfnisse der Werkstoffsuche zu lösen.

Trotzdem wurden verschiedene der gängigen Retrievalformen erst einmal unverändert getestet, um ihre Vor- und Nachteile - bezogen auf das Anwendungsfeld - zu klären. Dies geschah in drei Stufen:

- Realisierung eines lauffähigen Prototyps, z. B. im Modus Query by example (QBE).

- Empirische Tests mit Mitarbeitern der MTU, die später mit dem neu zu entwickelnden Werkstoffinformationssystem arbeiten sollen.

  Alle Testpersonen nutzen extensiv die Anwendungssoftware ihres Arbeitsgebiets, haben jedoch keinerlei softwareergonomische Kenntnisse.

- Auswertung der Tests mit dem Ziel, spezielle Vor- und Nachteile der einzelnen Modi zu erkennen.

Das Ergebnis dieser ersten Runde des rapid prototyping Zyklus war der Prototyp WING-M1, der versuchte, die Vorteile der einzelnen Modi zu integrieren und ihre Nachteile zu vermeiden. Z. B. wurde der QBE-Modus auf eine Veränderung der Ergebnistabellen beschränkt, da die Tests - auch aus anderen Untersuchungen bekannte - konzeptuelle Schwierigkeiten mit dem Join-Operator aufzeigten.

Ansonsten wurde der WING-M1 Prototyp auf „traditionelle Weise“ entwickelt, d. h., wie es die Styleguides und Kenntnisse kommerziell verfügbarer graphisch-direktmanipulativer Programmpakete nahelegen (cf. die Kurzbeschreibung von WING-M1 in Krause et al. 1993/94 und Wolff 1996).

Für unsere Zwecke ist es nicht nötig, näher auf WING-M1 einzugehen. Es genügt, die Bedienschwierigkeiten zu erläutern, deren Auflösung Ausgangspunkt für die Entwicklung der WOB-Strategien waren. Von ihnen wird angenommen, daß sie für eine größere Gruppe von Anwendungen auch außerhalb der Werkstoffbereichs gelten.

3.2 Bedienschwierigkeiten als Ausgangspunkt für die Entwicklung des WOB-Modells

Die für die Entwicklung des WOB-Modells relevanten Schwierigkeiten, die sich zum Teil bereits in früheren empirischen Studien gezeigt hatten, betreffen drei Hauptgruppen:

- Aus anderen Untersuchungen bereits bekannte inhärente Widersprüche zwischen softwareergonomischen Forderungen wie der nach hoher Selbsterklärungsfähigkeit für Anfänger und einem schnellen Bedienweg für Fortgeschrittene.

- Die in WING-IIR versuchte Modalitätsmischung von natürlicher Sprache als Abfragemodus und graphisch-direktmanipulativem Modus, sowie die

eingeschränkte Wirksamkeit beider Modalitäten in Situationen, in denen die Suchergebnisse als Liniengraphiken präsentiert werden.

- Schwierigkeiten, die sich dem Objekt-Funktions-Paradigma heutiger graphisch-direktmanipulativer Systeme zuordnen lassen bzw. generell die Frage nach der „richtigen" Art der Objektorientierung aufwerfen.

  Objekt-Funktions-Schema bedeutet, daß zuerst ein Objekt markiert, und danach über das Menü oder eine Aktionstaste die Handlung explizit benannt wird, die auf dieses Objekt wirken soll (cf. hierzu Abschnitt 4.3).

3.2.1 Rücknahme redundanter Selbsterklärungsfähigkeit vs. Erklärungsbedarf

Dieser altbekannte, inhärente softwareergonomische Widerspruch zeigte sich auch bei den WING-Tests. Es gibt Benutzer, die bei einem vorgegebenen Design eine größere Selbsterklärungsfähigkeit verlangen (z. B. zusätzliche Kommentartexte, Auswahlboxen statt Klapplisten usw.) und andere Benutzer, die in der gleichen Nutzungssituation eine Reduzierung der in ihren Augen redundanten Selbsterklärungsfähigkeit fordern, um z. B. mehr Platz für die Ergebnisdarstellung zu gewinnen.

Dieser Widerspruch wird oft an der Differenzierung Anfänger vs. Fortgeschrittene festgemacht. Die Konsequenz sind nicht selten zwei Modi der Bedienung, einer mit hoher Selbsterklärungsfähigkeit (meist verbunden mit größerem Platzverbrauch) und ein Modus, der Systemkenntnisse voraussetzt.

Solch eine Lösung funktioniert jedoch nicht. Die Einteilung in Anfänger und Fortgeschrittene spiegelt nur Maximalpunkte eines Kontinuums wieder. Die Übergänge sind fließend und zudem auf eine Weise benutzerabhängig, die sich nicht durch das Merkmal „Vorwissen zur Applikation" operationalisieren läßt.

Notwendig ist eine Lösung, die es erlaubt, benutzerindividuell und kontinuierlich die Selbsterklärungsfähigkeit des Systems zurückzunehmen. Diese Rücknahme darf zudem nicht Selbstzweck sein. Sie muß einen Gewinn bei anderen Bedienfaktoren erbringen, z. B. mehr Platz für die Ergebnisdarstellung zur Verfügung stellen oder die Anzahl der Bedienschritte durch Informationsverdichtung reduzieren (Anordnung auf einem statt zwei sequentiell hintereinander geschalteten Fenstern).

3.2.2 Flexibilität bei der Navigation vs. vorgegebene Leitlinien

Auch dieser inhärente softwareergonomische Widerspruch ist altbekannt und wurde vor allem bei hierarchischen Systemen immer wieder beobachtet. Einerseits wollen die Benutzer nicht darüber nachdenken, welcher Schritt als nächstes auszuführen ist, andererseits lehnen sie eine vorgegebene Abfolge als einengend ab, auch wenn sich dadurch alle Möglichkeiten der benötigten Suchkombinationen abdecken lassen.

Die erste Forderung würde man durch eine vorgegebene Navigationsleitlinie erfüllen: Einzelnen Teilschritten, die sich aus der Aufgabenanalyse ergeben, werden einzelne Fenster zugeordnet und die Abfolge der Fensteraktionen in eine möglichst „natürliche" Reihenfolge gebracht. Aber auch hier gibt es in der Regel keine benutzerübergreifenden Abfolgemuster. Verschiedene Benutzer ordnen die Teilschritte anders an, ohne daß sich hierfür im Einzelfall benutzer- oder nutzungsspezifische Parameter angeben und operationalisieren ließen.

„Natürlich" ist die potentielle Vielfalt der Abarbeitungsreihenfolge, nicht eine einzelne - nach welchen Kriterien auch immer festgelegte - Leitlinie.

Dies scheint zunächst relativ unproblematisch, da die Objektorientierung heutiger graphischer BOF diese Flexibilität unterstützt. Die Freiheit der Auswahl und das Setzen auf die selbstbestimmte Auslösung des nächsten Systemzustandes steht jedoch im Widerspruch zu der gleichzeitig gegebenen Benutzerforderung, möglichst ohne eigene Überlegungen einen „effizienten" und „sinnvollen" Weg durch den Aufbau der Suchanfrage gewiesen zu bekommen.

Auch hier kommt es zu der Situation, daß bei gleicher BOF, einige Benutzer mehr Flexibilität, andere eine stärkere Führung durch das System fordern.

Wie in Abschnitt 3.2.1 muß eine Lösung gefunden werden, die den Ausgleich zwischen flexibler, benutzerbestimmter Navigation und vorgegebenen Leitlinien benutzerindividuell und kontinuierlich vollzieht.

3.2.3 Iteratives Retrieval und „Denken" in Liniengraphiken, Skizzen etc.

Bereits bei einer Studie zu Patentinformationssystemen (cf. Krause/Womser-Hacker 1990) war aufgefallen, daß die Patentprüfer zwar erst einmal mit Deskriptoren in der Patentdatenbank suchten, die weitere Diskussion, ob ein

Dokument als Entgegenhaltung für ein neu eingereichtes Patent geeignet sei, jedoch oft anhand der Konstruktionsskizze führten. Sie artikulierten ihre weitere Suchabsicht dadurch, daß sie auf Bestandteile des Konstruktionsplans Bezug nahmen.

Das gleiche Phänomen tritt bei der Werkstoffsuche auf. Liegt ein erstes Ergebnis vor, das sich der Benutzer als Liniengraphik anzeigen läßt, „denkt" der Benutzer mit dieser Kurve weiter: „Das ist schon ganz schön, aber die Kurve des Elastizitätsmoduls müßte im niederen Temperaturbereich etwas höher liegen".

Das bedeutet, daß die Liniengraphik selbst als Modus zur Formulierung der weiteren Suchabsicht benutzt wird. Der Zwang, die Suchabsicht wieder als Parametermenge in der strukturierten Umgebung des graphisch-direktmanipulativen Ersteinstiegs zu formulieren, ist in dieser Situation genauso „unnatürlich" wie der natürlichsprachliche Abfragemodus. Der Modus der Ergebnisdarstellung bestimmt die für die Weiterführung der Suche optimale Modalität.

3.2.4 Verzicht auf explizite Handlungsanweisungen und mehrere Teilaufgaben in einer Dialogbox

WING-M1 verzichtete bereits auf Menüs, da sich schon bei früheren empirischen Studien im Bereich der Bürosoftware Schwierigkeiten bei der Bedienung ergeben hatten (cf. Mittermaier 1995, Krause 1993b). Nicht verzichtet wurde jedoch auf die Verwendung verbaler Handlungsanweisungen auf Aktionstasten (z. B. Umschlüsseln, Schließen) und auf die sequentielle Abarbeitung mehrerer Teilaufgaben in einer Dialogbox. WING-M1 enthält z. B. innerhalb eines Fensters Aktionsfolgen wie „Werkstoff eingeben / Taste 'Kennwert auswählen' drücken -> Folge: Liste der Kennwertausprägungen erscheint / Kennwertausprägung markieren / Taste 'Suche starten' -> Ergebnisbildschirm".

Aber auch in diesen Bereichen zeigten sich deutliche Schwierigkeiten:

- Handlungsanweisungen auf den Aktionstasten wurden falsch interpretiert und verwirrten.
- Mußten in einer Dialogbox mehrere Teilaufgaben durchlaufen und einzeln abgeschlossen werden, stieg die Fehlerquote.

Diese Erfahrungen führten zu einem Designentwurf, der das durch die Styleguides nahegelegte, weitverbreitete Objekt-Funktions-Schema der Interaktion weitgehend vermeidet.

3.2.4.1 Menüs

Aus dem Zeitraum 1987 bis Ende 1993 liegen eine Reihe empirischer Studien zu komplexen Textverarbeitungssystemen wie COMFOTEX, WORD und N/JOY vor (cf. Mittermaier 1995, Krause 1993b, Weingärtner 1992). Allein mit COMFOTEX wurden über 100 Benutzer mit verschiedenen Menüvarianten getestet. Es zeigte sich wiederholt, daß auf die Menüs ein Großteil der Bedienschwierigkeiten entfällt. Auch nach längerer Eingewöhnungszeit gelingt es Benutzern nicht problemlos, Handlungsoptionen im Menü aufzufinden, richtig anzuwenden und die darin enthaltenen Elemente der Zustandsanzeige adäquat zu interpretieren. Unter anderem sind hierfür die Benennung der Menüeinträge, die Gruppenbildung und die Zustandsanzeige durch den Häkchenmechanismus verantwortlich. Dieses Ergebnis überraschte um so mehr, als Menüs zu den am besten erforschten Komponenten graphischer Systeme gehören (cf. Paap/Roske-Hofstrand 1988 als Überblick).

Am Häkchenformalismus wird zudem deutlich, daß heutige Menüs bereits eine Mischung zweier Modalitäten repräsentieren: natürliche Sprache (Optionsbenennungen) und graphisch-bildliche Gestaltung (Strukturierung, Häkchen und Objekte als Ikonen). Bereits im Mikrokosmos des Menüsubsystems läßt sich zeigen, welche Schwierigkeiten der weitere Ausbau multimedialer Systeme für die softwareergonomische Analyse erwarten läßt (die Häkchenproblematik bleibt hier ausgeklammert, cf. Krause 1993b).

a) Breite, Tiefe, Gruppenbildung und Bezeichnungsvielfalt von Menüsystemen

   Die folgende Tabelle gibt einen Einblick in die Bezeichnungsvielfalt, die Gruppenbildung sowie in die Breite und Tiefe komplexer Menüs, wie sie bereits zur Zeit der empirischen Tests zur COMFOWARE erreicht war. Die heutigen Systeme wie WORD für WINDOWS95 erweiterten diese Komplexität nochmals.

   Die Gesamtanzahl der Einträge in der Menüleiste, den Pull-down und den Kaskaden-Menüs zeigt das Grundproblem: Der Funktionsumfang wird zu groß, um von einem Menüsystem bewältigt zu werden. So fordert z. B. der Siemens Nixdorf Styleguide (1992) - im wesentlichen im Einklang

mit der softwareergonomischen Literatur - die folgenden Obergrenzen nicht zu überschreiten:

(1) Menüleiste (1. Ebene) maximal 9 Einträge

(2) Pulldown-Menü (2. Ebene) max. 12 Elemente und maximal 7 Gruppen

(3) Maximal Dreistufigkeit. Da die Dialogboxen in die hierarchische Stufung einbezogen werden müssen, bedeutet dies den Verzicht auf eine zusätzliche Stufe im Pulldown-Menü (= Kaskadenmenü als 2. Ebene).

Generell versuchen die Softwareentwickler diese Regeln möglichst zu befolgen. Daß dies nicht mehr gelingen kann, ist aber bei allen Applikationen nur eine Frage der Versionsnummer. Die softwareergonomischen Regeln wirken derzeit hauptsächlich in Richtung einer Vertiefung des hierarchischen Systems, entweder über die Einführung einer zusätzlichen Menüebene (Kaskaden-Menü) wie bei AMIPRO (8 Kaskaden-Menüs mit 34 zusätzlichen Elementen) oder versteckter über Dialogboxen wie bei WORD (teilweise dreifach gestaffelt). Die empirischen Tests zu COMFOTEX ergaben jedoch gerade für tiefe Hierarchien die größten Schwierigkeiten für den Benutzer.

Herkömmliche Menüs überschreiten bei funktionsreichen Applikationen zwangsläufig die Grenzen softwareergonomisch optimaler Gestaltung. Ihre Vorteile (Gruppenbildung durch Hierarchisierung und Vorlage der Handlungsoptionen statt Merkleistung) lassen sich nicht mehr voll ausspielen. Sie werden durch die Suchproblematik und die langen Bedienwege ins Gegenteil verkehrt.

b) „Natürliche“ Benennung und Hierarchisierung

Folgt man den Styleguides und der softwareergonomischen Literatur (cf. Helander 1988, Eberleh et al. 1994, GSM 1995), hätten Entwickler von Menüs eine relativ einfache Aufgabe: Sie befragen Benutzer, wie sie einzelne Anweisungen und Objekte benennen und in welchem hierarchischen Verhältnis die Terme zueinander stehen, und verwenden beides bei der Menügestaltung.

Unglücklicherweise sieht die Realität anders aus. Im Rahmen der wissenschaftlichen Begleitforschung zur COMFOWARE wurden verschiedene Versionen von Menüs für die COMFOTEX-Textverarbeitung entwickelt. Die Grundproblematik blieb jedoch die gleiche: Benutzer arbeiteten nicht

effizient mit den Menüs. Sie verbrachten viel unnütze Zeit mit dem Öffnen der falschen Pull-down Menüs und interpretierten einzelne Funktionen unzutreffend.

Vergleicht man zudem den Wechsel der Bezeichnungen zwischen einzelnen Versionen der großen graphischen Textverarbeitungsprogramme WORD, AMIPRO, WORDPERFECT, WORDSTAR und COMFOTEX und die Differenzen zwischen den konkurrierenden Produkten, zeigen sich die unterschiedlichsten Hierarchiebildungen und eine große Spannbreite der Benennungen.

Teilweise sind die Unterschiede der mangelhaften softwareergonomischen Durchdringung kommerzieller Software zuzuschreiben. Im Kern geht es jedoch um etwas anderes: Sprachliche Begriffe lassen sich in vielen Anwendungsbereichen in hierarchischen Begriffsnetzen anordnen, die bei der Mehrzahl der Benutzer übereinstimmen. Die Kategorisierung nach Ober- und Unterbegriffen ist eine mächtige kognitive Grundtechnik der Informationsverarbeitung (cf. Treu 1992, Durding et al. 1977, Roppel 1996: Abschnitt 1.3). Durchgehende hierarchische Strukturen sind jedoch nicht wie Booch 1992:12 übergeneralisierend feststellt, in allen Fällen eine Grundvoraussetzung für das Verstehen komplexer Systeme. Es gibt Bereiche und Begriffsgruppen, bei denen die Benutzer über keine „natürliche" Gliederung nach Ober- und Unterbegriffen verfügen und sich eine vom Entwickler konstruierte Hierarchisierung auch nicht ohne Lernaufwand durchsetzen läßt. Die Studien zu COMFOTEX ergaben deutliche Hinweise, daß zumindest Teile der Textverarbeitung zu letzterer Gruppe gehören. Man muß jedoch grundsätzlich bei allen Applikationen mit diesem Phänomen rechnen.

Fazit Menüs: Komplexe Menüs enthalten inhärente Schwachstellen, die in Anforderungen des Gegenstandsbereichs und in Regeln der sprachlichen Kategorisierung begründet sind. Ihre Auswirkungen lassen sich durch sorgfältige Gestaltung abmildern, aber nicht beseitigen. Menüs sind somit im Sinne der Softwareergonomie nur suboptiomal gestaltbar. Deshalb verzichtet schon der WING-M1 Entwurf weitgehend auf Menüs.

3.2.4.2 Handlungsoptionen auf Aktionstasten in WING-M1

Der weitgehende Verzicht auf Menüs in WING-M1 schloß allerdings Handlungsanweisungen wie „Umschlüssel" auf Aktionstasten nicht ein, die im Prinzip dem gleichen Objekt-Funktions-Schema unterliegen. Aktionstasten teilen allerdings nicht die Hierarchisierungsprobleme mit den Menüs, die bei

den COMFOTEX-Tests als hauptsächlicher Auslöser der Bedienschwierigkeiten erschienen.

Die empirischen Tests von WING-M1 zeigten jedoch, daß bei Handlungsoptionen auf Aktionstasten vermehrt Fehler auftraten. Die Benutzer hatten vor allem Schwierigkeiten, die Handlungsbezeichnungen richtig zu interpretieren und ihren Fokus zu bestimmen.

Die Konsequenz aus diesen erneuten Problemen war, nach einer Lösung zu suchen, die auf Handlungsbezeichnungen weitgehend verzichtet, womit auch das weitverbreitete Objekt-Funktions-Schema verlassen werden muß.

3.2.5 Modalitätsmischung zwischen natürlichsprachlicher Abfrage und graphisch-direktmanipulativer Gestaltung

Der Prototyp WING-M1 bot neben dem graphisch-direktmanipulativen Zugang ein permanent offenes Fenster an, in dem die Benutzer ihre Anfrage alternativ natürlichsprachlich formulieren konnten. Multimedialität von Sprache und graphisch-direktmanipulativem Modus bedeutete somit gleichzeitige Verfügbarkeit.

Die empirischen Tests zeigten, daß der natürlichsprachliche Modus kaum angenommen wurde (cf. Marx 1995). Andererseits kommentierten die Benutzer bei den Einzeltests mit dem natürlichsprachlichen Interface diese Modalität nicht generell negativ.

Welchen Grund gibt es für die konsequente Bevorzugung des graphisch-direktmanipulativen Modus?

Gegen die These, der natürlichsprachliche Modus sei generell ungeeignet, sprechen die Ergebnisse und Benutzerkommentare der Einzeltests. Deshalb scheint es sinnvoll, die Begründung in der Art der Modalitätsmischung zu suchen.

Fragt man sich, wie die beiden Modalitätsmischungen anders verbunden werden könnten, wird eine generelle Kenntnislücke im Bereich der Multimodalität deutlich. Softwareentwickler haben heute zwar die technische Möglichkeit, zumindest Text, graphische Elemente und eingeschränkt Ton und Sprache gleichzeitig einzusetzen. Wann welche Information mit welcher Modalität am effizientesten codiert wird, und welche Mischformen softwareergonomisch vorteilhaft sind, darüber gibt es jedoch kaum Erkenntnisse. Die-

se konzeptuelle Lücke läßt im extrem expansiven Bereich der multimedialen Systemsoftware ergonomisch unbefriedigende Lösungen erwarten, sobald man die immer wieder zitierten Einzelbeispiele offensichtlicher Mischformen wie z. B. Bild- und Tonhinterlegung bei Musiklexika oder die Verbindung von gesprochener Sprache mit Zeigegesten verläßt.

Im WOB-Modell wird deshalb versucht, zumindest für die Verbindung von natürlichsprachlicher und graphischer Interaktion eine Mischform zu realisieren, die die Vorzüge der einzelnen Modalitäten bewahrt und gleichzeitig versucht, die ihnen inhärenten Nachteile zu vermeiden.

Wie in Abschnitt 4.2.4 gezeigt wird, integriert die vorgeschlagene Lösung zudem die in Abschnitt 3.2.1 diskutierte Problematik der dynamischen Rücknahme von Selbsterklärungsfähigkeit.

3.2.6 Fazit Ergebnisse WING-M1 Tests

Gesucht wird somit ein Designentwurf, der die in den Abschnitten 3.2.1 - 3.2.5 diskutierten Widersprüche ausgleicht und die angesprochenen Probleme minimiert. Da es sich um generelle softwareergonomische Problemstellungen handelt, sollte die vorgeschlagene Lösung über den ursprünglichen Anwendungsbereich hinausweisen. Neben einigen generellen und bekannten softwareergonomischen Widersprüchen geht es vor allem um Lösungen für die Modalitätsmischung und um eine Auslegung der Objektorientierung, die nicht in Konflikt mit anderen Forderungen gerät.

Die Einzellösungen für die Probleme der Abschnitte 3.2.1 - 3.2.5 müssen dabei so miteinander verbunden werden, daß die Gesamtlösung die bestmögliche softwareergonomische BOF ergibt. Dieser Ausgleich verschiedener Forderungen ist der Hauptvorteil eines Designmodells auf einer - im Vergleich zu den konkreten Styleguideregeln auf der einen Seite und den allgemeinen Grundsätzen der ISO-Normen bzw. den kognitionspsychologischen Theorien auf der anderen - mittleren Abstraktionsebene.

4 Grundelemente des WOB-Modells

Die in Kap. 3 diskutierten Schwierigkeiten wurden in eine Reihe aufeinander abgestimmter Gestaltungsmaßnahmen umgesetzt, die in ihrer Gesamtheit die Basis des WOB-Modells bilden. Sie waren zum ersten Mal Grundlage des Prototypen WING-M2, der 1994 mit Benutzern der MTU getestet wurde. Die Akzeptanz lag deutlich über, die Fehlerquote unter der des WING-M1 Tests (cf. Krause/Womser-Hacker 1996).

Die folgende Tabelle gibt einen Überblick über die derzeit das WOB-Modell konstituierenden Prinzipien. Sie werden in den folgenden Abschnitten erläutert und begründet. Die weitergehende Visualisierung auf der Basis von „visual formalisms“ diskutiert Krause 1996 eingehender.

WOB - Modell

- **Generelle SE - Prinzipien**
  - **dynamische Anpassung / kontextsensitive Durchlässigkeit**
  - **Dialogleitlinien**
  - **intelligente Komponenten**
- **Iteratives Retrieval und Ergebnis → Suche → Transformation**
  - **Graphisches Ergebnisretrieval**
  - **Eingeschränkter Query by Example - Modus**
- **Zustandsanzeige mit Korrekturmodus**

↙	↘	↘
Natürlichsprachl. Generierung	**Komprimierter Einstiegsbildschirm**	**Formalsprachliche Generierung**

- **Strikte Objektorientierung und doppelte Interpretierbarkeit:**

  Werkzeugmetapher ⇦⇨ Formularmodus
- **Weitgehende Visualisierung auf der Basis von „visual formalisms“**

Gegenüber dem Ursprungsmodell 1994 kommt die weitgehende Visualisierung als eigenständiges Gestaltungsprinzip hinzu. Sie war bisher nur indirekt im Prinzip des iterativen Retrievals als Alternative enthalten. Diese Ausweitung bedeutet keine Modelländerung, sondern eine Ergänzung in Hinblick auf die erwartete Weiterentwicklung des WOB-Modells in der mittleren Entwicklungsperspektive. Die Integration weitgehender Visualisierungsformen in das WOB-Modell wird ausführlich in einem gesonderten Arbeitsbericht, in Krause 1996, behandelt.

Gleichzeitig wurde das Metaphernkonzept neu überdacht. Im überarbeiteten WOB-Modell bleibt das Formular nur noch „Erklärungsmetapher". Die ursprüngliche „Formularmetapher" wird als „Formularmodus" i.S. der „visual formalisms" - Debatte gesehen. Visual formalisms sind im Kern nichtbildhafte, nichtmetaphorische, visuelle Gestaltungsmittel wie Tabellen, spread sheets, Liniengraphiken oder hierarchische Baumstrukturen, deren graphischer Charakter in Verbindung mit den kognitiven Grundfähigkeiten des Menschen (Raumwahrnehmung u.a.) eine effiziente, direktmanipulative Systembedienung ohne (bzw. mit nur geringem) Lernaufwand ermöglichen und die Problemlösung durch „external representation" unterstützen. Im WOB-Modell geht es nicht um eine Entweder-Oder-Enscheidung zwischen Metaphernbildung und visual formalisms, wie dies Nardi/Zarmer 1993 suggerieren. Beide Gestaltungstechniken werden miteinander verbunden und jeweils dort eingesetzt, wo sie Vorteile versprechen (zur Begründung cf. Krause 1996).

4.1 Generelle softwareergonomische Prinzipien: Dynamische Anpassung und kontextsensitive Durchlässigkeit

Aus der Gruppe der generellen softwareergonomischen Prinzipien werden nur die dynamische Anpassung und die kontextsensitive Durchlässigkeit behandelt. Das Prinzip der Dialogleitlinien greift Abschnitt 4.2.3 im Kontext der doppelten Interpretierbarkeit auf. Eine Diskussion intelligenter Komponenten enthalten Krause et al. 1993/94 und Krause/Womser-Hacker 1996.

Die Zielsetzung der Integration von Formen einer dynamischer Anpassung ist zweifach:

- Vermeidung redundanter Eingaben durch intelligentes Systemverhalten

  Wird eine Eingabe in einem Feld an anderer Stelle nochmals benötigt, soll der Benutzer nicht gezwungen werden, die Information ein zweites Mal

einzugeben. Unerheblich muß es dabei sein, ob das gleiche Fenster davon betroffen ist oder einige Bildschirme zwischen den Feldern liegen. In beiden Fällen lösen Benutzereingaben im aktuellen Fenster selbständig Anpassungsvorgänge in anderen Fenstern bzw. innerhalb eines Fensters aus. Unerheblich soll auch sein, ob direkte Identität der Eingabe besteht oder beide Felder durch eine deduktive Ableitung miteinander verbunden sind. Im letzteren Fall muß das System z. B. erst „folgern", was der Oberbegriff zu einer Werkstoffbezeichnung ist, die im ersten Feld eingegeben wurde.

Das System weiß somit prinzipiell über den vorausgegangenen Dialog innerhalb einer Aufgabenlösung Bescheid.

In der ursprünglichen Formulierung des WOB-Modells von 1994 wurden die „dynamische Anpassung" und „kontextsensitive Durchlässigkeit" als verschiedene Prinzipien voneinander getrennt. Zuordnungskriterium war dabei, ob die von der Anpassung betroffenen Felder im gleichen Fenster liegen. Diese Differenzierung scheint jedoch eher untergeordneter Natur zu sein. Beide Terme beschreiben den gleichen Sachverhalt, nur aus verschiedenen Blickwinkeln, und sollten deshalb auch nicht fachsprachlich für spezielle Unterklassifikationen differenziert werden.

- Effiziente Platzverwendung

  Benutzereingaben in einem Feld lösen Anpassungsvorgänge in anderen Listen- bzw. Auswahlfeldern aus. Liegen z. B. im Verbandsinformationssystem ELVIRA zum Thema Außenhandelsdaten für England keine Werte vor, verkürzt sich der Länderbrowser um diesen Eintrag.

  Dieser kontextsensitiven Anpassung von Listen entspricht eine dynamische Anpassung von Controlgrößen. So erweitert sich im Verbandsinformationssystem ELVIRA das Feld für die Eingabe der Länder jeweils genau um eine Leerzeile, nachdem der Benutzer ein Land direkt eingegeben oder aus der Länderliste ausgewählt hat.

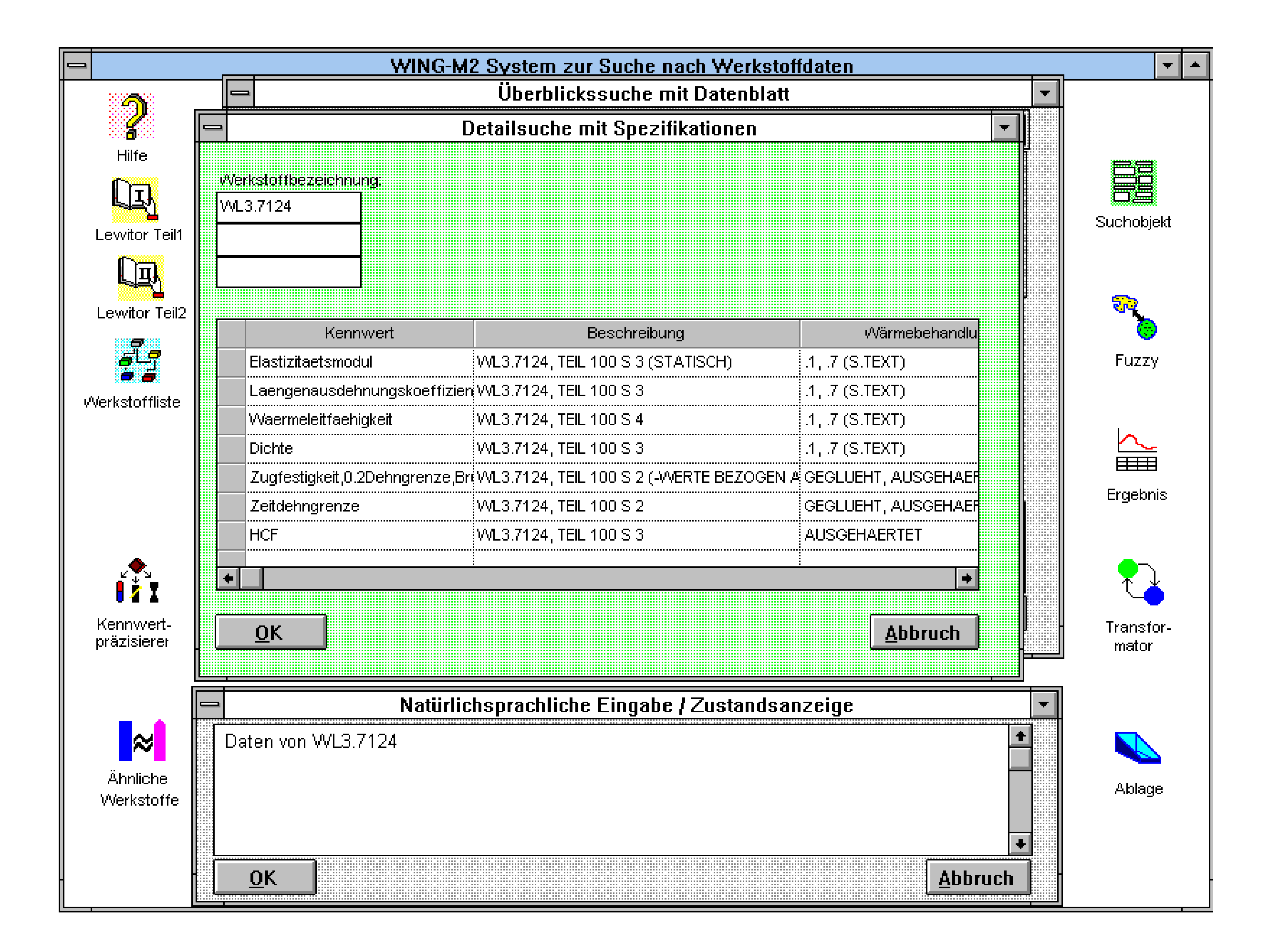
  Die dynamische Anpassung von Feldgrößen und Listeninhalten erbringt nur dann den Effekt der effizienteren Platzverwendung, wenn eine bestimmte Komplexitätsgrenze überschritten wird. Passen alle sinnvollen Handlungsalternativen auf einen Bildschirm, hat eine dynamische Anpassung von Listeninhalten oder Auswahlfeldern nur einen negativen Effekt. Die Ortsfestigkeit der Einträge und der Vorlagecharakter aller Möglichkeiten, Parameter zu setzen, geht verloren. In diesem Fall sind Grausetzun-

gen der in einem spezifischen Kontext nicht möglichen Alternativen die bessere Lösung.

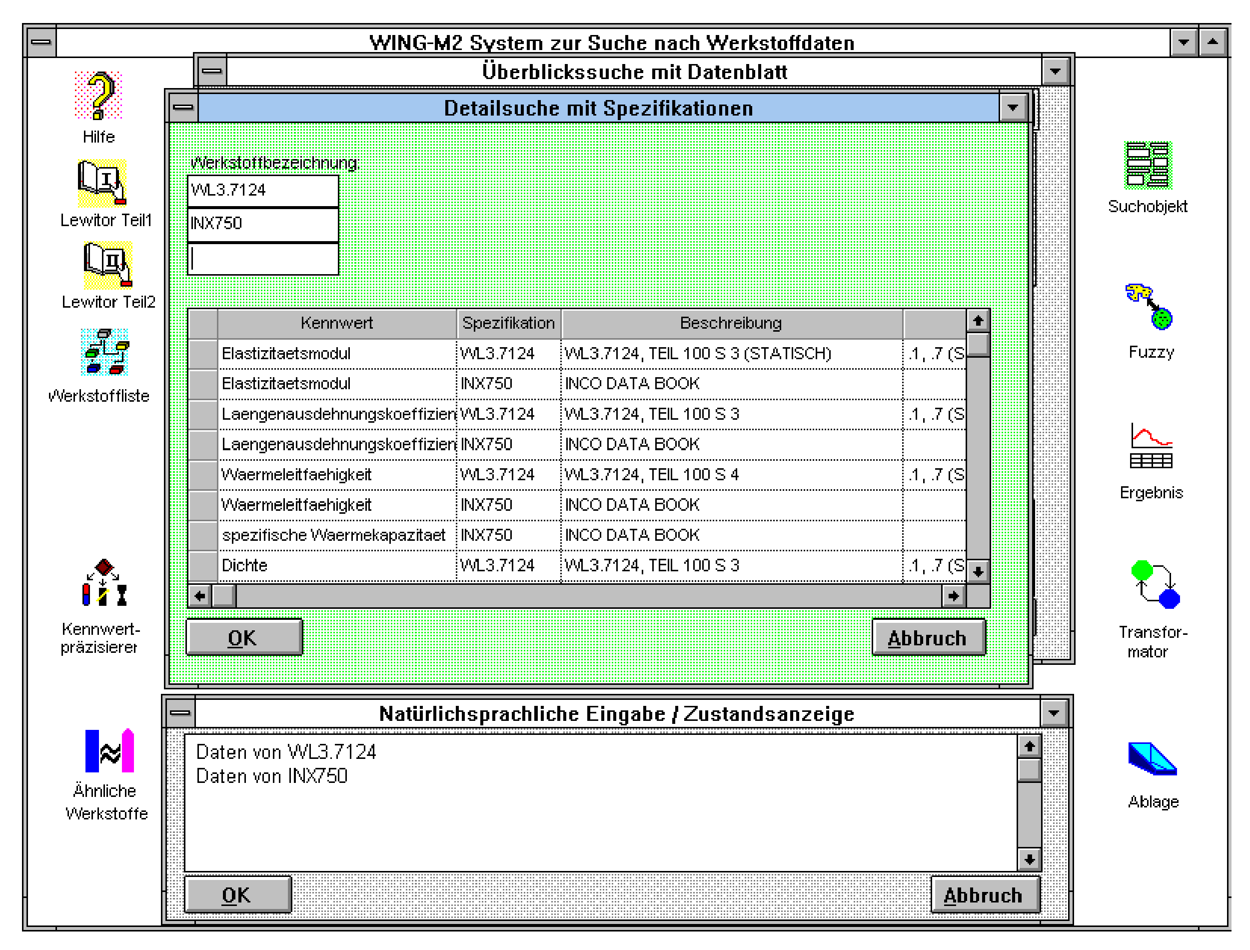
Bei einer hoher Komplexität der Anwendung wertet das WOB-Modell den Platzgewinn jedoch in der Regel höher als die Ortskonsistenz.

Beispiel 1: Listenanpassung innerhalb eines Fensters

Die dynamische Anpassung der Feldinhalte innerhalb eines Fensters begegnete bei WING-M2 auch dem Problem der höheren Fehleranfälligkeit bei mehreren Teilaufgaben in einem Fenster und teilweise den Schwierigkeiten mit Handlungsanweisungen auf Aktionstasten (cf. Abschnitt 3.2.4).



In WING-M2 paßt sich die Kennwertliste, sobald eine Werkstoffbezeichnung eingegebenen wird, dynamisch an die aktuell möglichen Kennwerte an (Vereinigungsmenge bei zwei oder drei Werkstoffen).



Die sequentielle Folge von WING-M1 Teilschritten aus dem in Abschnitt 3.2.4 angeführten Beispiel, die der Benutzer explizit weiterschalten muß, „Werkstoff eingeben / Taste 'Kennwert auswählen' drücken → Folge: Liste der Kennwertausprägungen erscheint (im gleichen Fenster) / Kennwertausprägung markieren / Taste 'Suche starten' → Ergebnisbildschirm“, wurde durch die automatische Anpassung der Kennwertliste nach Eingabe der Werkstoffbezeichnung ersetzt.

Beispiel 2: Kontextsensitive Durchlässigkeit über Bildschirmgrenzen hinweg

WING-M2 differenziert am Eröffnungsbildschirm u.a. zwischen zwei Suchfenstern, die systemseitig geöffnet sind:

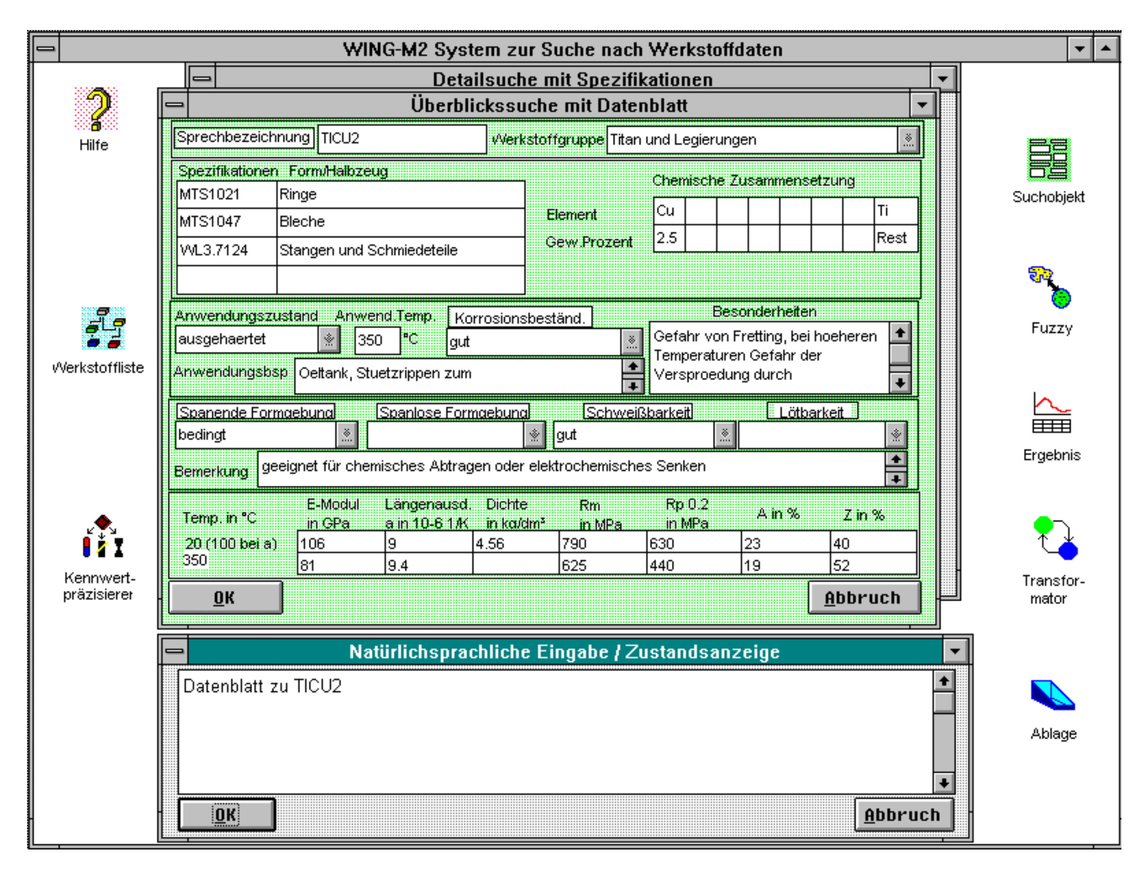
- Detailsuche mit Spezifikationen (im folgenden als Spezifikationssuche bezeichnet; cf. die Bildschirme von Beispiel 1).

  Spezifikationen sind einzelne Werkstoffe mit bestimmten Eigenschaften (z. B. einer bestimmten Form), zu denen Messungen durchgeführt wurden.

- Überblickssuche mit Sprechbezeichnungen im Datenblatt (Standard: geöffnet und bis auf die Überschriftszeile durch das aktive Fenster der Spezifikationssuche überdeckt).

  Sprechbezeichnungen sind Oberbegriffe von Spezifikationen. Die hier angegebenen Parameter sind größtenteils Durchschnittswerte aller Spezifikationen, die zur Werkstoffsprechbezeichnung gehören, bzw. gelten sie für alle Spezifikationen gleichermaßen. Die Spezifikationen der Sprechbezeichnung sind als Liste im Datenblatt angegeben.

  Für diesen Suchtyp hat sich als Kurzform der Term 'Datenblattsuche' eingebürgert, der im folgenden benutzt wird.



Beide Suchfenster differenzieren somit nach detaillierten Informationen zu einer Werkstoffspezifikation versus generellen Informationen zum Werkstoff, der im Datenblatt als Oberbegriff (der Sprechbezeichnung) fungiert.

Die beiden gestapelt übereinander liegenden Fenster 'Spezifische Suche' und 'Datenblattsuche' sind als zwei Sichtweisen (views) der gleichen Suchanfrage zu interpretieren, die kontextsensitiv durchlässig gestaltet werden. Das System ermittelt z. B. zur Spezifikation WL 3.7124 (erster Bildschirm in Abschnitt 4.1.1) automatisch den Werkstoff „TICU2“ (Sprechbezeichnung) im Datenblatt. Damit kann der Benutzer ohne zusätzliche redundante Eingaben zum generellen Aspekt seiner Suchanfrage überwechseln, dort fortfahren oder seine spezielle Suche weiter spezifizieren.

Beim Beispiel 2 realisiert das System die kontextsensitive Durchlässigkeit (bzw. dynamische Anpassung) ohne Benutzereingriff, da die inhaltlichen Beziehungen faktisch eindeutig sind.

Daneben gibt es eine zweite Form, bei der der Benutzer die Durchlässigkeit steuert:

Markiert ein Benutzer Spezifikationen einer Ergebnisliste und aktiviert danach das Suchfenster ‘Spezifikationssuche’, werden die markierten Elemente als Parameter in das Feld für die Werkstoffspezifikationen übernommen und können somit die Grundlage einer neuen Suche bilden.

Das Markierungskonzept gilt auch für die Durchlässigkeit zwischen der 'Datenblattsuche' und 'Spezifikationssuche'. Eine im Datenblatt vom Benutzer markierte Spezifikation wird in das Fenster der Spezifikationssuche übertragen. Diese benutzerseitige Steuerung der Durchlässigkeit durch Markierung ergänzt die im Hintergrund ablaufende permanente Übernahme aller eindeutig übertragbaren Parameter.

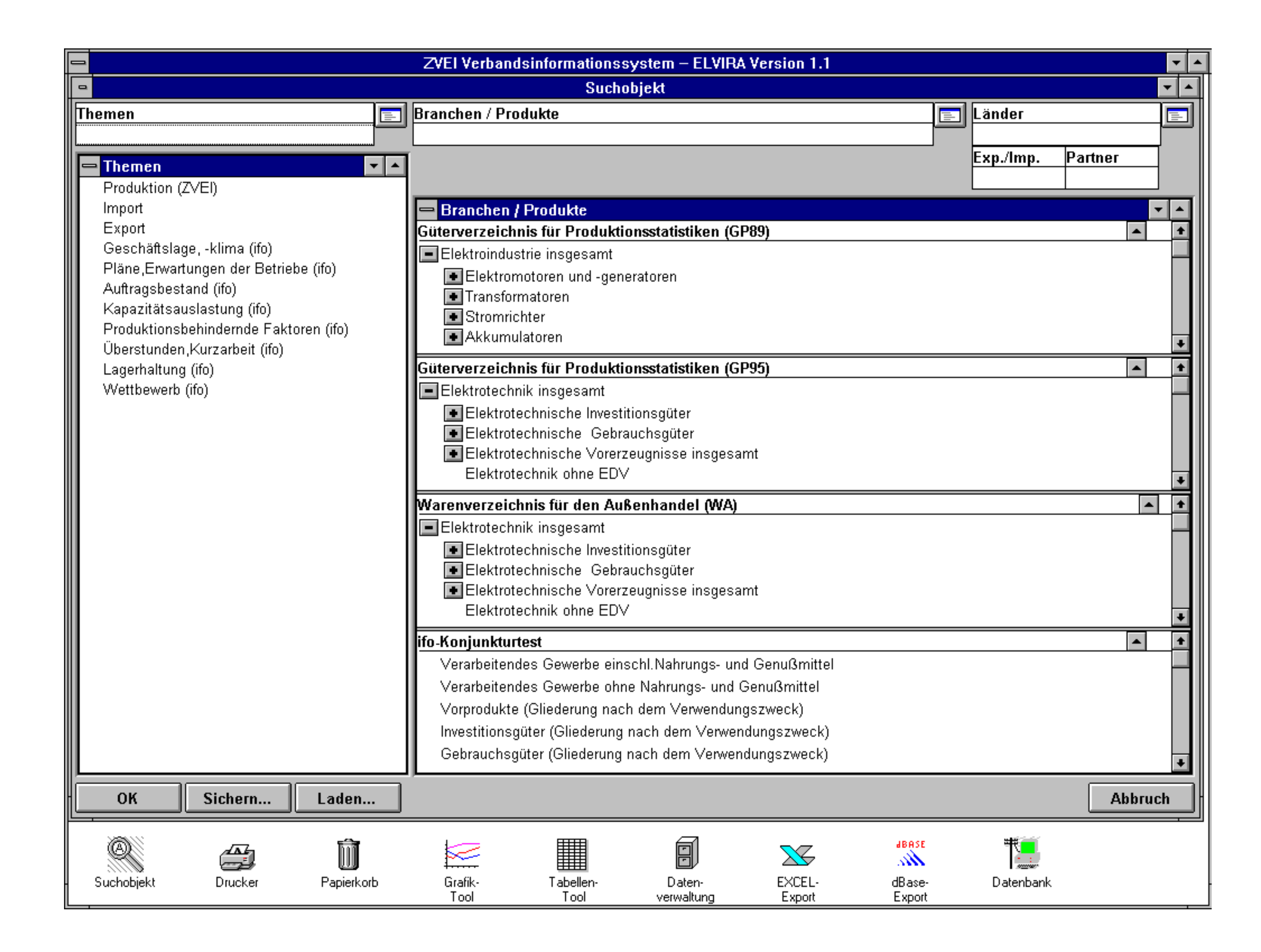
Beispiel 3: Feldübertragung und dynamische Anpassung von Feldgrößen im Suchobjekt von ELVIRA

ELVIRA enthält wirtschaftliche Faktendaten (nationale Produktionsstatistik, Außenhandelsstatistik und Konjunkturtestdaten), die als Zeitreihen in tabellarischer Form oder als Liniengraphiken ausgegeben werden.

Jede Zeitreihe wird durch Deskriptoren beschrieben, die sich einer von drei Facetten zuordnen lassen: den Themen (z. B. Produktionsdaten), den Branchen/Produkten (z. B. elektrotechnische Gebrauchsgüter) und den Ländern.

Eine typische Anfrage wäre: „Gib mir die Zeitreihen zu den Produktionsdaten für Transformatoren in Deutschland und England“.

ELVIRA bietet die Möglichkeit, Deskriptoren sowohl direkt oder mit Hilfe von drei Browsern zu den Facetten einzugeben. Die Felder zur Direkteingabe bilden gleichzeitig die Zustandsanzeige, in die auch Terme aus den Browsern durch Anklicken übertragen werden. Vor allem der Browser für die Branchen/Produkte enthält eine Fülle von hierarchisch geordneten Einträgen.



Der vom Benutzer gewünschte Browserausschnitt ist in der Regel größer als der zur Verfügung stehende Bildschirmplatz. Um trotzdem die Suchformulierung auf einer Bildschirmseite in einem Schritt zu ermöglichen (ohne durch

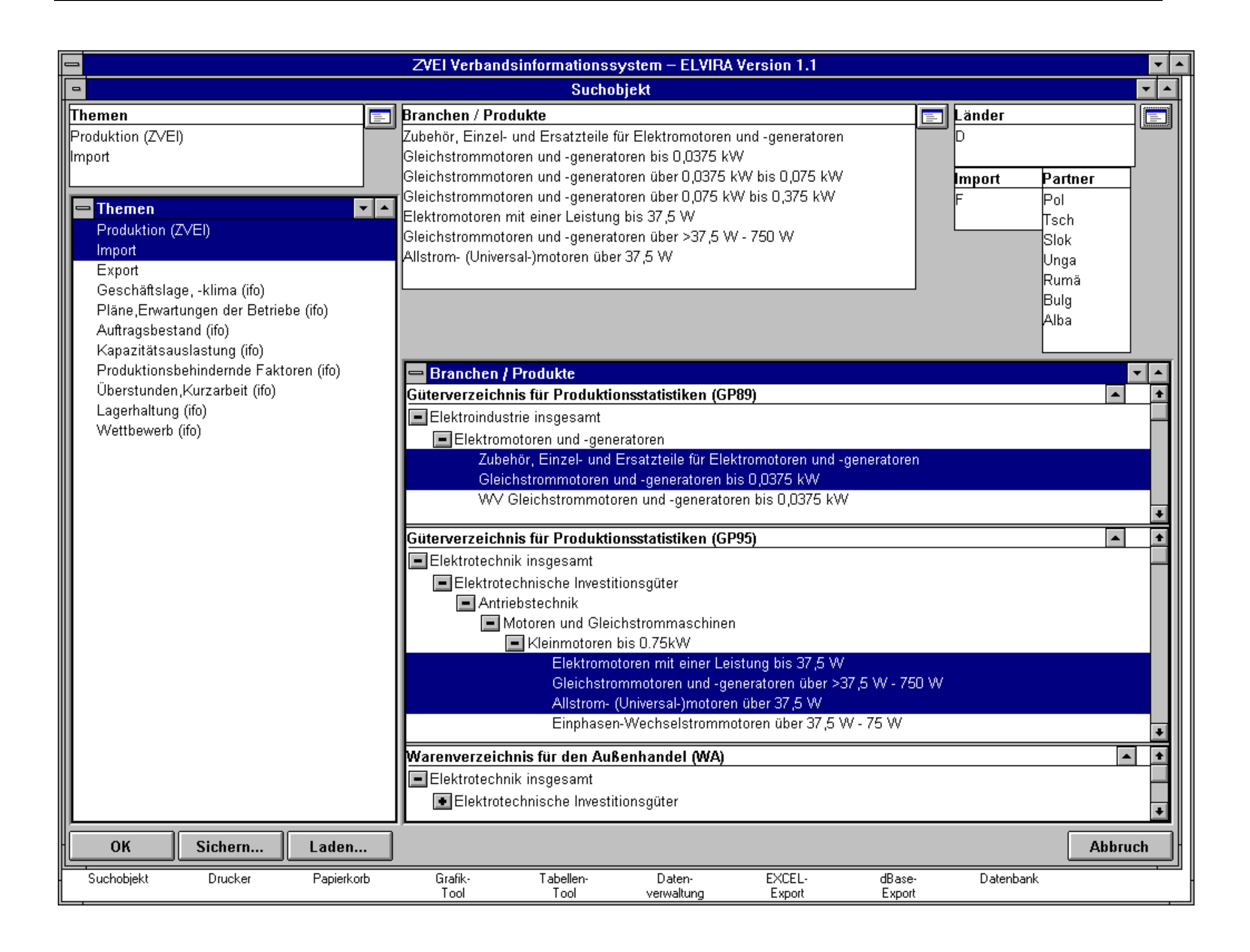
überlagernde Fenster verdeckte Auswahlinformationen), wird die dynamische Anpassung der Eingabefelder und Browserfenster extensiv genutzt.

- Jeder Eintrag in eine Facette verändert die Browserinhalte der beiden anderen. Sind Außenhandelsdaten z. B. nur für Deutschland vorhanden, wird Deutschland in der Länderfacette vorbelegt. Der Länderbrowser enthält keinen weiteren Eintrag.

- Die Felder zur direkten Eingabe lassen immer genau eine Zeile frei. Das Feld wächst dynamisch nach jedem Deskriptoreneintrag. Gleichzeitig paßt sich der darunterliegende Browser an den restlichen Platz an, so daß er immer den maximal möglichen Raum einnimmt (keine Leerflächen).

- Für die Branchen/Produkte - Facette sind nebeneinander verschiedene Schlagwortsysteme im Gebrauch. Z. B. wurde der Suchbaum ab 1995 umgestellt, wodurch es nebeneinander den Suchbaum GP89 und GP95 als Güterverzeichnis für Produktionsstatistiken gibt. Ein vom Benutzer eingegebener Term kann somit in mehr als einem Unterverzeichnis enthalten sein. In diesem Fall stellt sich der Browser dynamisch auf beide Unterverzeichnisse ein.

  Die Grenze zwischen den beiden Verzeichnissen läßt sich vom Benutzer frei verschieben, wenn er z. B. mehr Kontext für das erste Unterverzeichnis wünscht, wobei der gewählte Deskriptor immer sichtbar bleibt (automatische Fokusanpassung).



Das Prinzip der dynamischen Anpassung hat in all seinen Möglichkeiten, die die Beispiele 1 - 3 exemplarisch zeigen, jeweils für sich genommen softwareergonomische Vorteile. Einen Großteil seiner Wirkungen entfaltet sich jedoch erst in Verbindung mit den folgenden Gestaltungsregeln des WOB-Modells.

4.2 BOF und Objektorientierung

Für ein Verständnis der Grundideen des WOB-Modells ist der Gebrauch des weitverbreiteten Terms Objektorientierung zu klären. Ausgehend von der objektorientierten Programmierung beherrscht er heute auch die Diskussion um die Oberflächengestaltung graphisch-direktmanipulativer Systeme, ohne daß allerdings klar wäre, was präzise damit gemeint sein soll. In der Realität überlagern sich - auch innerhalb der Styleguides - verschiedene Auslegungen (cf. Rosson/Alpert 1990, Cohen 1991, Winblad et al. 1990, Quibeldey-Circel 1994).

Da die „richtige“ Interpretation von Objektorientierung - nicht im technischen, sondern im konzeptuellen Sinn - für alle Überlegungen zum WOB-Modell entscheidend ist, kann an dieser Stelle auf eine grundsätzliche Diskussion der Spielarten und Möglichkeiten objektorientierter BOF nicht verzichtet werden.

Der Begriff der Objektorientierung ist am klarsten bei der objektorientierten Programmierung (cf. Meyer 1990, Sager 1991, Quibeldey-Circel 1994) und (eingeschränkter) bei den objektorientierten Datenbanken definiert (cf. Cox/Novobil-ski 1991). Bei BOF wird er heute im wesentlichen als zugkräftiges Schlagwort benutzt, ohne daß schon klar wäre, was Objektorientierung von BOF eigentlich bedeutet. Deshalb liegt es nahe, bei der Klärung des Begriffs von seiner Verwendung im Bereich der objektorientierten Programmierung als Fixpunkt auszugehen.

Im Kern ist Objektorientierung eine Sichtweise auf die Ereignisse und Zustände der Welt. Sie wird aus der Perspektive der Objekte gesehen. Von ihnen aus strukturiert der Benutzer seine Aufgabe; ihnen ordnet er auch die Handlungen zu, die er mit ihnen ausführt. Im Gegensatz hierzu stehen z. B. bei kommandoorientierten Programmier- und Dialogsprachen Handlungsbezeichnungen ('Lösche' ...) im Zentrum, denen Objekte als Spezifikationsklassen zugeordnet werden.

Graphische BOF legen die objektorientierte Sichtweise nahe. Die Metaphorik ist in der Regel so gewählt, daß die Ikonen physikalische Objekte der Realität abbilden (z. B. die Schränke und Ordner der Bürometapher für die Dateiablage), auf die direktmanipulativ zugegriffen werden kann. Gleichzeitig haben Applikationen, die objektorientiert programmiert sind, eine starke Affinität zu graphischen Oberflächen.

Die Objekte der graphischen BOF stehen in einer Reihe von Beziehungen, von denen in unserem Kontext vor allem die Generalisierung bzw. Spezialisierung und die Klassifikation bzw. Instanziierung interessieren. Die damit möglichen Hierarchisierungen und Klassenbildungen (mit Merkmalsvererbung) sind ein wesentliches Kennzeichen aller objektorientierten Systeme.

Vergleicht man die objektorientierte Programmierung mit objektorientierten BOF, zeigt sich, daß sich die Beurteilungskriterien verschieben. Die oben genannten Beziehungstypen erlauben z. B. in der objektorientierten Programmierung das leichte Einfügen neuer Objekte in bestehende Strukturen. Alles was das neue Objekt mit bestehenden gemeinsam hat, muß nicht neu programmiert werden, sondern steht über die Klassenzuweisung und Vererbung problemlos und zudem übersichtlich zur Verfügung.

Bewertet man objektorientierte BOF per se (entkoppelt vom Entstehungsprozeß) sind diese Beurteilungsgründe (die für die Entwicklung der Oberflächen wichtig sind) vom softwareergonomischen Standpunkt her irrelevant. Hier wird als entscheidend angesehen, daß Hierarchisierung und Klassenbildung konzeptuell natürliche Vorgehensweisen sind, über die Menschen als allgemeine kognitive Strategien verfügen. Bildet man sie in der elektronischen Welt ab, führt dies zu einer erhöhten Selbsterklärungsfähigkeit und effizienteren Bedienung. Der Benutzer muß seine eigene Sichtweise auf die Welt nicht in eine EDV-motivierte transformieren (cf. aber die Einschränkungen in den folgenden Abschnitten).

Überträgt man das Konzept und die Definitionen der Objektorientierung von der Programmierung in die softwareergonomische Oberflächengestaltung, hofft man natürlich, daß die definitorischen Merkmale auch unter den dort gültigen Entscheidungskriterien positiv interpretiert werden können (wie im obigen Beispiel der Hierarchisierung und Klassenbildung). Selbstverständlich liegt dies bei der Objektorientierung schon deshalb nahe, weil auch Programmierer Benutzer sind, deren kognitiv-natürliche Denkweisen die Güte und Effizienz des Programmierprozesses beeinflussen. Trotzdem muß der Nutzen der Objektorientierung bei jedem einzelnen Merkmal der Übertragung nachgewiesen werden. Die Gefahr besteht darin, eine Vorgehensweise aus der objektorientierten Programmierung unhinterfragt zu übernehmen, die z. B. durch Veränderungseffizienz beim 'software cycle' gerechtfertigt wird (was den Benutzer beim Bedienen der Oberfläche nicht betrifft), aber keiner kognitiv natürlichen Benutzerkonzeptualisierung entspricht. Ein weiterer Gefahrenpunkt ist, daß bei vielen graphischen Applikationen Anfänger und gelegentliche Benutzer im Vordergrund stehen, wäh-

rend beim Programmieren von erheblichen Lernzeiten ausgegangen werden muß (auch wenn sie sich gegenüber früher immer stärker verkürzen). Arbeiten Entwickler mit objektorientierten Programmierwerkzeugen, kommt noch hinzu, daß sie die hier gültigen Konzeptualisierungen nach einer bestimmter Zeit der Praxis auch dann verinnerlichen, wenn sie im Widerspruch zur natürlichen kognitiven Vorgehensweise stehen. Verwenden sie diese Prinzipien bei der Oberflächengestaltung, kommen sie selbst und ihre Kollegen damit sehr gut zurecht, da sie auf ihr Altwissen aus der Praxis der objektorientierten Programmierung zurückgreifen. Sie merken nicht mehr, daß für Benutzer ohne diese Vorkenntnisse Bedienhemmnisse aufgebaut werden.

Diese Überlegungen sprechen nicht gegen eine Anwendung der objektorientierten Sichtweise auf BOF, auch nicht gegen den Versuch, das sich im Kontext der objektorientierten Programmierung relativ stabil herausgebildete Paradigma zu übernehmen. Sie warnen jedoch vor einer ungeprüften, nicht hinterfragenden Kopie.

Es spricht viel dafür, daß im Detail Objektorientierung bei BOF anders modelliert werden muß als bei der objektorientierten Programmierung oder bei objektorientierten Datenbanken, da sich die Bewertungsfaktoren zwar partiell überschneiden, aber im Einzelfall unterschiedlich sind (cf. das obige Beispiel der Veränderungseffizienz).

4.2.1 Funktionstasten vs. Meldungen an Objekte und Objektorientierung in der natürlichen Sprache

Bei dem Versuch, die Prinzipien der objektorientierten Programmierung in Kriterien zur Gestaltung von BOF umzusetzen, gibt es kaum Probleme bei der Klassenbildung und Hierarchisierung und auch nicht bei der Kapselung, die verlangt, daß jedem Objekt alle Funktionen (= die Methoden) zuzuordnen sind, die es ausführen bzw. von denen es betroffen werden kann. Damit sind die Methoden auch untereinander abgegrenzt. Sie stehen nur ihrem - und keinem anderen - Objekt zur Verfügung.

Probleme der Konzeptualisierung gibt es bei der Aktivierung der Methoden.

> „ Methoden sind der Verarbeitungsteil in den Objekten. Sie bestimmen das Verhalten der Objekte. ... Auch Methoden definiert man in der Klasse. Sie werden durch Nachrichten, die ein Objekt erhält, aktiviert ... In einem Objekt sind damit alle Daten und die Funktionen, die auf diese Daten wirken, zusammengefaßt (Kapselung). ... Methoden bestehen wie

> Prozeduren und Funktionen in konventionellen Sprachen aus Anweisungsfolgen.“ (Sager 1991:38)

Daraus müßte - denkt man diese Konzeptualisierung konsequent zu Ende - z. B. in der Textverarbeitung folgen, daß ein Text die Methode, kursiv gesetzt zu werden, inhärent als Methode in sich hat. Soll ein Text kursiv gesetzt werden, bekommt das Objekt 'Text' die Nachricht zugeschickt, die ihm inhärente Methode 'kursiv setzen' auszulösen.

Ein anderes Beispiel wäre ein Signal in einem Bahnhof (cf. das Beispiel in Sager 1991), das die Fähigkeit in sich trägt, eine Zugdurchfahrt zu stoppen (durch die Signalstellung und eine automatische Notbremsvorrichtung, falls das Signal trotzdem passiert wird).

Beim Signalbeispiel haben wir mit der Konzeptualisierung wenig Schwierigkeiten. Auch in der Umgangssprache konzeptualisieren wir in dem Sinne, daß das Signal als technische Vorrichtung über bestimmte Handlungsmöglichkeiten verfügt, die es auslöst, wenn es eine bestimmte Nachricht bekommt Signalstellung ändern, Bremsschuh hochfahren usw. Bei der Textverarbeitung können wir die Konzeptualisierung abstrakt zwar noch mitvollziehen, es drängt sich jedoch eine andere „natürliche“ Konzeptualisierung auf: Mit dem Text geschieht etwas; er wird umgewandelt. Der Text selbst hat bei dieser Sichtweise somit die Fähigkeit, kursiv zu werden, nicht inhärent in sich, sondern es handelt sich um eine Aktion, die von außen einwirkt (nicht nur als Nachricht).

Noch deutlicher wird der Unterschied zu einer Konzeptualisierung der Welt durch Objekte mit inhärenten Methoden, die durch Nachrichten aktiviert werden, und der Konzeptualisierung, die Benutzer durch ihren Umgang mit der natürlichen Sprache als „natürlich“ empfinden, wenn wir versuchen, einen Satz wie „Hans ermordet Fritz“ in der Sichtweise der objektorientierten Programmierung auszudrücken.

Fritz müßte hier - als Objekt der Welt - die Methoden „ermordet werden“ inhärent in sich tragen. Die Zustandsänderung in der Welt, daß Fritz ermordet wurde, käme durch eine Nachricht an Fritz zustande, die fordert, die Methode „ermordet werden“ auszulösen.

Diese Denkweise müßte dem Benutzer bei einer 1:1 Übertragung der Konzeptualisierung, die die objektorientierte Programmierung anwendet, nahegelegt werden. Daß sie ihm so fremdartig erscheint, hat mit der Art zu tun, in der der Mensch bei der Verwendung der natürlichen Sprache die Welt sieht.

Das Beispiel zeigt, daß der von Quibeldey-Circel 1994:150 postulierte direkte Zusammenhang zwischen der Art der Konzeptbildung in der sprachlichen Erfassung der Welt und der der objektorientierten Programmierung nicht besteht.

> „Das Potential des Objekt-Paradigmas liegt ... in der linguistischen Parallele. Nach der Hypothese von Benjamin Whorf prägt die Sprache unser Denken. --- Wenn nun die technische Sprache der Systemanalytiker und des Programmierers auch eine konzeptionelle (und nicht nur syntaktische) Teilmenge der natürlichen ist, dann trägt diese Einschließung auch zur Verinnerlichung des Objekt-Paradigmas bei. ... Die Lerndisposition für objektorientierte Konzepte ist durch die natürliche Sprachfähigkeit mitgegeben. ... Die objektorientierten Begriffe verlieren ihre Mystik: Sie sind der natürlichen Sprache vertraut“ (Quibeldey-Circel 1994:149f.).

Dies ist ein Mißverständnis. Richtig ist nur, daß es nahe liegt, bei der Suche nach einer kognitiv natürlichen Konzeptualisierung von BOF die der natürlichen Sprache zu untersuchen. Entscheidet man sich für eine Konzeptualisierung parallel zu der natürlichen Sprache, muß man nicht von der sprachwissenschaftlichen Whorf-These ausgehen, daß Menschen prinzipiell sprachlich gebunden denken. Für die softwareergonomische Gestaltung genügt die Tatsache, daß Menschen die sprachliche Grundkonzeptualisierung täglich anwenden und einüben. Sie steht somit jedem Benutzer als Altwissen, als „natürliche“ Sichtweise auf die Welt zur Verfügung.

> „Die weitaus meisten einfachen Sätze unserer Sprache haben Subjekt-Prädikat- (Objekt-) Struktur, d. h. sie bestehen aus Eigennamen, die für bestimmte Objekte ... stehen, und aus einem Prädikat, das diesen Objekten ein Attribut zuschreibt (seien es Eigenschaften, Beziehungen, Zustände, Prozesse oder Handlungen). Dabei werden die Objekte aufgefaßt als mehr oder minder dauerhafte Gegebenheiten, die durch ihre verschiedenen momentanen Zustände hindurch im Wechsel ihrer Attribute ihre Identität bewahren, ... an denen sich das Geschehen vollzieht und die als Träger der Zustände fungieren. In diesem Sinn interpretieren wir gewöhnlich alle Ereignisse und Sachverhalte in einer Objekt-Attribut-Struktur.“ (Kutschera 1971:308f.)

Daß es sich hier um eine kognitive Konzeptualisierung (bei Kutschera eine Interpretation der Welt) handelt, die weder selbstverständlich noch ontologisch vorgegeben ist, zeigt sich z. B. am Japanischen. Hier fehlt wie

bei allen Sprachen des ostasiatischen Raums diese Art der „Objektorientierung „.

> „Die Beziehung einer Handlung zu dem sie ausführenden Tätersubjekt ist für abendländisches Denken so wichtig, daß es uns schwer fällt, ein Denken nachzuvollziehen, in dem gerade dieser Bezug einmal nicht als unbedingter Tatzusammenhang gesehen und zum anderen für so sekundär gehalten wird, daß er unerwähnt bleiben kann.“ (Hartmann 1952:71)
>
> ... daß diese für uns mit dem Begriff des Sehens notwendig verbundene Vorstellung eines Subjekts im Japanischen in allen begrifflichen Abarten des Sehens fehlt, diese vielmehr unpersönlich und tätersubjektlos sind - eine Tatsache, die auch für alle anderen Vorgangsbezeichnungen im weitesten Sinne gilt.“ (Hartmann 1952:19)

Ähnliche Abweichungen hat Whorf 1956 für die Sprachen der Hopi und Nootka festgestellt. In den indogermanischen Sprachen wird dagegen vom Objekt-Funktions-Schema mit seiner Einteilung in aktive (handelnde) und passive (betroffene) Objekte nur in ganz wenigen Einzelfällen abgewichen (Geschehensverben und Ausdrücke wie „es regnet“), die in der Linguistik starke Interpretationsschwierigkeiten bereiten.

Die Überlegungen zeigen, daß die direkte Übernahme der Konzeptbildung nach dem Muster der objektorientierten Programmierung, die Quibeldey-Circel 1994 mit Objektorientierung gleichsetzt, bei bestimmten Anwendungskonstellationen auf Schwierigkeiten stößt. Um sie bei BOF auch in bezug auf die Methoden (Funktionen / Aktionen) sinnvoll anwenden zu können, die in der natürlichen Sprache passiven (leidenden) Objekten zuzuordnen sind, müßte sie als spezielle Sichtweise auf die Welt gelernt und eingeübt werden, bevor das Konzept seine Vorteile entfalten kann. Dies erscheint bei der Programmierung keine ernstliche Hemmschwelle für die Verwendung, wohl aber bei graphisch-direktmanipulativen BOF.

Insofern bietet sich das Objekt-Funktions-Schema, wie es in vielen Systemen - als „schwache“ Auslegung der Objektorientierung realisiert ist, durchaus als eine „natürliche“ Art der Konzeptbildung an, die durch ihre Entsprechung in der natürlichen Sprache nahegelegt wird.

Fassen wir die Wirkungsweise des Objekt-Funktions-Schemas nochmals zusammen: Entsprechend der objektorientierten Sichtweise geht der Benutzer von einem Objekt aus und markiert es (z. B. ein spezifisches Dokument; Leitfrage: Mit was will ich als nächstes etwas tun?). In einem zweiten Schritt (somit nachgeordnet im Unterschied zu kommandoorientierten Dialogsprachen) sucht er sich die Handlungsoption (Leitfrage: Was will ich mit dem

ausgewählten Objekt tun?). Hierzu hat er drei Möglichkeiten: Tastenkombination, Menü und Toolboxen.

Die Bedienung über Tastenkombination und Toolboxen wird im folgenden ausgeklammert, um die Darstellung nicht zu überfrachten.

Im Menü sind die Handlungsoptionen verbal symbolisiert und in einer hierarchischen Struktur mehrstufig geordnet. Das Problem, daß einzelne Handlungen nicht global wirken, sondern nur auf bestimmte Objekte, wird durch Graufärbung der Einträge und durch wechselnde Menüeinträge gelöst.

Der wesentliche Unterschied zur Konzeptbildung in der objektorientierten Programmierung ist, daß die direkt benannten Funktionen (Methoden) quasi gleichberechtigt mit den Objekten die Sichtweise auf die Welt bestimmen. Deshalb scheint es berechtigt, bei dieser Art der Objektorientierung von objektbasierten BOF zu sprechen, bzw. von einer „schwachen“ Auslegung der Objektorientierung, die gegen eine „strikte“ Objektorientierung abgegrenzt wird. Bei „strikter Objektorientierung“ soll ein Objekt-Objekt-Schema als verbindlich für die Konzeptbildung vorgeschrieben sein.

Gegen eine Übernahme des Objekt-Funktions-Schemas in das WOB-Modell sprechen die in Abschnitt 3.2.4 genannten Schwierigkeiten aus den empirischen Tests: In WING-M1 führten die Handlungsoptionen auf den Funktionstasten vermehrt zu Fehlern und bei den COMFOTEX-Tests hatten die Benutzer Probleme, Einträge in komplexen Menüs korrekt zu interpretieren.

Deshalb wird - trotz der prinzipiell positiven Einschätzung - nach anderen Möglichkeiten der Auslegung einer objektorientierten BOF gesucht.

4.2.2 Funktionsobjekte und drag & drop

Neben dem Objekt-Funktions Muster gibt es in den Styleguides und vielen kommerziellen Produkten ein zweites, dem eine andere Art der Objektorientierung zugrunde liegt, die Funktionsobjekte. Idealiter ergibt sich ein einfaches Objekt-Objekt-Schema, dessen prinzipielle Unterschiede zum Objekt-Funktions-Schema herausgearbeitet werden sollen.

a) Der Benutzer entscheidet sich für das Objekt, das er bearbeiten will, und markiert es.

b) Er sucht sich ein zweites Objekt („aktives Werkzeug“), aktiviert es als Funktionsobjekt und wendet es auf das in Schritt a) markierte Objekt (das „passive“) an. Das Standardverfahren für die Zuweisung ist das „drag & drop“-Prinzip.

   Wesentlich für die These der globalen Verwendung von Funktionsobjekten ist, daß sie die Menüs ersetzen sollen. Darum muß die Handlung dem Werkzeug eindeutig inhärent sein.

   Z. B. tut ein Reißwolf nichts anderes als Objekte zu zerstören. Deshalb ist es überflüssig, die Handlung „Löschen“ explizit zu benennen.

Die mögliche Metapher hinter den Funktionsobjekten sind einfache Handlungen mit einfunktionalen Werkzeugen in der Welt. Die generelle Grundidee soll im folgenden tentativ an 'Hans ermordet Fritz' durchgespielt und dabei gleichzeitig verfremdet werden.

In der Sichtweise der Funktionsobjekte sucht sich Hans zuerst Fritz als Referenzobjekt seiner Handlung aus, überdenkt dann, welches Werkzeug er verwenden soll, z. B. ein Messer, und wendet das Messer dann auf Fritz an, indem er es dazu bringt, die ihm inhärente Handlungsweise auszuführen, z. B. eine lebenswichtige Ader zu durchtrennen.

Dies klingt erst einmal fremdartig. Trotzdem wissen wir, daß es Beispiele für Funktionsobjekte gibt, die einem ganz natürlich vorkommen.

In Büroverwaltungssystemen wie COMFODESK ist z. B. das Löschen mit dem Reißwolf enthalten: Der Benutzer sucht sich das zu zerstörende Objekt aus und markiert es (z. B. ein Textdokument). Er ergreift es mit Hilfe der Maus, schiebt es auf den Reißwolf und läßt es los, wodurch das Dokument zerstört wird.

Weitere Vorgänge, die bei Beibehaltung des Objekt-Objekt-Schemas und „drag & drop“ problemlos mit der natürlichen Vorgehensweise in der realen Welt parallelisiert werden können, sind alle Formen des Ablegens eines Objekts in einem anderen (Objekt-Objekt-Beziehung, kein Objekt-Funktionsobjekt-Paar). Will der Benutzer einen Brief in einem Ordner ablegen, entscheidet er sich zuerst für das Objekt, das er ablegen möchte, ergreift es mit der Maus und läßt es über dem zweiten Objekt, dem Ordner, los, wodurch es in den Ordner „fällt“ bzw. eingeordnet wird.

Beiden als stimmig empfundenen Beispielen ist gemeinsam, daß sich die drag & drop - Handlung mit einer lokalen Bewegung vom passiven zum aktiven Objekt - in Übereinstimmung mit der Handlung in der realen Welt - in Beziehung setzen läßt. Hinzu kommt eine eineindeutige Verbindung einer ganz

bestimmten Handlung mit dem aktiven Objekt (dem Funktionsobjekt als Werkzeug). Beim Messer im ersten Beispiel ist das nicht so. Messer kann man zu den verschiedensten Handlungen einsetzen.

Analysiert man Produkte wie N/JOY, die auf Menüs weitgehend verzichten, erweist sich der Gegensatz Funktionsobjekt vs. Menüsystem als Kern des Anspruch, die „einzig echten“ objektorientierten Oberflächen realisiert zu haben. Die Funktionsobjekte müssen Menüeinträge hier über die genannten positiven Beispiele hinaus ersetzen. Eine modifizierte Form der Funktionsobjekte tritt vollständig an die Stelle einer Objekt-Funktions-Zuordnung über Objektmarkierung und expliziter Funktionsauslösung mit Hilfe des Menüs. Alle Handlungsoptionen müssen als inhärente Merkmale von Objekten modelliert werden.

Aus einem derartigen Oberflächenkonzept ergeben sich eine Vielzahl von Gestaltungsproblemen, die Weingärtner 1992 auf der Basis eines empirischen Tests mit N/JOY ausführlicher abhandelt. Hier soll die Problematik nur an einigen Teilbereichen exemplifiziert werden.

Allein die zu gewährleistende Funktionalitätsbreite macht das Bereitstellen aller möglichen (individuellen) Objekte einer Klasse - schon aus Platzgründen - nicht möglich. So können z. B. nicht sämtliche verfügbare Schriftarten als Stifte (= Objekt für Schriftart) auf der Oberfläche angeboten werden. Aus diesem Dilemma mußte N/JOY einen Ausweg finden und hat dazu das Dialogboxkonzept (in abgeänderter Form) übernommen: Die Objekte haben vom System vordefinierte Eigenschaften, die bei Bedarf über eine Dialogbox verändert werden können. Das Werkzeug wird mit Hilfe der Dialogbox „eingestellt“; es ist multifunktional geworden. Wie bei einem Druckkugelschreiber die Farben, kann man im funktional stark erweiterten N/JOY-Bleistift zusätzlich die Größe, Schriftart usw. wechseln und damit die Bleistifte beliebig vermehren. Dieses Verfahren scheint eine sinnvolle, und vom Benutzer tolerierbare Erweiterung der Realweltmetapher.

Kritischer ist eine andere Abweichung von den Erfahrungen in der realen Welt. Der Benutzer markiert eine Textstelle, die er mit einer bestimmten Schrift ergänzen möchte. Dazu wird das Werkzeug des Stiftes 'Helvetica 12' gewählt. Bleibt man beim drag & drop - Prinzip, müßte der Benutzer jetzt das Teildokument auf den Bleistift ziehen. In der realen Welt geht das natürlich gerade umgekehrt. Die Richtung stimmt nicht mehr. Deshalb geht N/JOY auf das abstraktere Prinzip des Markierens des passiven Objekts mit anschließender Aktivierung des zweiten (aktiven) Objekts (Werkzeugs) über und

verwendet hierzu wiederum abstrakt bestimmte Mausmodi (Differenzierung in rechte und linke Maustaste).

Das Abgehen von dem einfachen Interaktionsschema der zu Beginn von Abschnitt 4.2.1 besprochenen eindeutigen Funktionsobjekte wie dem Reißwolf „Markieren betroffenes Objekt + drag & drop zum Funktionsobjekt“ wird aber noch aus einem anderem Grund zwingend.

N/JOY kennt keine Typendifferenzierung zwischen Funktionsobjekten und den durch Funktionsobjekte betroffenen. Es gibt nur Objekte: Ein Kopierer kopiert einen Papierkorb; der Papierkorb läßt ein Dokument verschwinden usw.

Ein direkter Vorteil dieses Verfahrens ist, daß die Zuordnung des jeweiligen Status von der Zielstellung abhängt („betroffenes“ oder „handelndes“ Objekt). Er ist vom Benutzer zielabhängig in der aktuellen Anwendungssituation festzulegen. Für den Benutzer besteht damit jedoch eine potentielle Ambiguität: Ist ein Objekt in der aktuellen Dialogsituation nun ein „handelndes“ Objekt (Funktionsobjekt) oder eine „betroffenes“? Ein Beispiel: Ein Drucker druckt (mit dem Objekt PRINTER ist also die Funktion „Drucken“ verknüpft); ein Drucker kann aber auch kopiert werden. Der Drucker wird damit zum Objekt der Funktion „Kopieren“.

Auf dieser Ebene dürften beide Handlungsmöglichkeiten noch relativ einsichtig und klar differenzierbar sein.

Schwierig wird es für den Benutzer von N/JOY, wenn ein Objekt, das aus Benutzersicht keinen unmittelbaren Funktionscharakter hat, wie etwa eine Position im Dokument, plötzlich Funktionscharakter übernehmen muß. Ein Beispiel: Ein Graphik-Bereich ist an einer bestimmten Stelle im Dokument einzufügen. Hierzu muß das Objekt AREA-GRAPHIC markiert und im Anschluß daran die gewünschte Position im Dokument aktiviert werden.

Grundsätzlich übernehmen in N/JOY Objekte die Aufgabe, Handlungsalternativen anzubieten: Sie repräsentieren das Funktionsangebot. Die Objekte stehen jedoch nicht von vornherein in Beziehung zu einem anderen Objekt oder sind in den Kontext der aktuellen Dialogsituation eingebunden.

Die Folge ist eine sehr komplexe Mausbedienung, die bereits für sich genommen eine weitere wichtige Ursache für die bei den empirischen Tests festgestellten Bedienhemmnisse war.

a) Die Funktionsauslösung, im N/JOY-Kontext genauer die Aktivierung einer Funktion, erfolgt generell mittels der linken Maustaste: Eine Funktion wird durch einfaches Anklicken des entsprechenden Objekts ausgelöst, oder ein Objekt, auf das eine Funktion angewandt werden soll, mittels der gedrückten linken Maustaste über das mit der gewünschten Funktion verknüpften Objekt gezogen.

b) Da das Öffnen einer zu einem Objekt gehörenden Dialogbox, nicht in die unmittelbare Funktionsanwendung integrierbar ist, braucht N/JOY hierfür eine davon unabhängige Methode: Über die zweimalige Betätigung der rechten Maustaste sind generell die Dialogboxen von Objekten zugänglich; die Container-Objekte dagegen mit zweimaligem Anklicken der linken Maustaste.

c) Objekte können in N/JOY aus Benutzersicht sowohl Objektstatus wie auch Funktionsstatus einnehmen. Je nach Status muß ein Objekt markiert oder aktiviert werden. Eine derartige Konzeption erfordert natürlich ebenfalls unterschiedliche Handlungsschritte im Benutzerdialog: Zur Markierung eines Objekts, dem damit aus Benutzersicht Objektstatus zukommen soll, muß die rechte Maustaste eingesetzt werden (einfach anklicken). Auch zur Aufhebung der Markierung ist die rechte Taste erforderlich.

   Die Anwendung eines Objekts, was aus Benutzersicht als Zuweisung des Funktionsstatus an ein Objekt beschrieben werden kann, erfolgt dagegen mit der linken Maustaste.

N/JOY verlangt somit zur Bedienung beide Maustasten mit jeweils zwei Klickoperationen (einfaches oder doppeltes Anklicken) und dadurch vom Benutzer einiges an Übung und vor allem ein hohes Maß an Konzentration im Umgang mit der Maus. Ein Benutzer muß klare Vorstellungen über sein Ziel haben und entsprechend bewußt die linke oder die rechte Maustaste einsetzen. Daneben muß er immer bedenken, ob und was nun eigentlich markiert ist, bevor er die linke Maustaste drückt. Beachtet er dies nicht ausreichend, kann dies fatale Folgen haben.

Die Maus als zentrales Eingabeinstrument graphisch-direktmanipulativer Oberflächen soll dem Benutzer eigentlich die Arbeitsweise erleichtern und Tastaturkommandos ersetzen. Auf das Objekt zu zeigen, mit dem man arbeiten will, ist natürlicher und steht dem menschlichen Handeln näher als komplizierte Kommandos zu lernen und sie über die Tastatur einzugeben. N/JOY verspielt diesen Anspruch auf leichte Handhabbarkeit durch den generell notwendigen Einsatz beider Maustasten in jeweils zwei Modi und die abstrakte Maustastenauslösung.

Insgesamt gesehen kann das Konzept der erweiterten, multifunktionalen Funktionsobjekte nicht mehr intuitiv mit Rekurs auf das Handeln in der realen Welt und die natürliche kognitive Konzeptualisierung interpretiert werden. Es wird zu dem, was es auch in der objektorientierten Programmierung ist: ein zwar im intellektuellen Aufbau einfaches, klares und effizientes, aber theoretisch-abstraktes Konzept zur Auslösung von Handlungen. Um es sinnvoll anwenden zu können, muß die objektorientierte Sichtweise auf die Welt nach dem Muster der Funktionsobjekte eingeübt und die ebenfalls abstrakten Auslösemechanismen für das Markieren, Aktivieren und Öffnen von Objekten gelernt werden.

4.2.3 Fazit

Das theoretisch-abstrakte Verfahren des Arbeitens mit Funktionsobjekten läßt sich bei einigen Fällen mit dem Vorgehen in der realen Welt in Einklang bringen, nicht jedoch überall. Dort, wo es metaphorisch „natürliche“ Entsprechungen gibt, können Funktionsobjekte problemlos eingesetzt werden. Geht man über den Bereich hinaus, verzichtet man auf den Rekurs auf Vorerfahrung des Benutzers und damit auf die intuitive Erschließbarkeit des Verfahrens.

Empirische Tests mit N/JOY (cf. Weingärtner 1992) bestätigten, daß Benutzer das dort praktizierte Objekt-Objekt-Schema nicht mehr beherrschen. Auch strikte Objektorientierung führt somit für sich genommen nicht zu natürlich bedienbaren BOF.

Funktionsobjekte lassen sich nur dann problemlos einsetzen, wenn das Bild des (Ein)legens/Einschiebens des Objekts auf/in das Funktionsobjekt in der realen Welt eine Parallele hat und hohe Eindeutigkeit der Funktionszuordnung gegeben ist (in diesem Sinn positiv: Drucken, Dokument in Schrank ablegen, Dokument auf Kopierer ziehen usw.). Zudem muß die Richtung der Aktion stimmen. Auch in der realen Welt soll das Objekt auf das Funktionsobjekt zu bewegt werden und nicht umgekehrt (z. B. kein Funktionsobjekt Schreibstift). In diesen engen Grenzen erwiesen sich Funktionsobjekte bei den empirischen Tests als ein wesentlicher Bestandteil direktmanipulativer BOF, die die Selbsterklärungsfähigkeit erhöhen.

Funktionsobjekte können somit die Objekt-Funktions-Bedienung nicht vollständig ersetzen, wie dies N/JOY versucht. Sie sind als ergänzende Strategie sinnvoll in einem eingeschränktem Bereich, in dem sich das theoretisch-abstrakte Objekt-Objekt-Schema mit den Vorgängen in der realen Welt parallelisieren läßt. Hier eingesetzt erhöhen sie den direktmanipulativen Charakter des Gesamtsystems.

Die N/JOY-Lösung, durch eine verstärkte Objektorientierung im Sinne multifunktionaler Funktionsobjekte als Menüersatz das Objekt-Objekt-Schema global einzuführen, hat sich zumindest in der hier versuchten Art nicht bewährt und wurde deshalb auch nicht in das WOB-Modell integriert.

Damit scheiden erst einmal beide der bisher vorgeschlagenen Hauptvarianten der Interpretation von Objektorientierung für eine direkte Übernahme in das WOB-Modell aus. Bevor der Grundgedanke der Werkzeugmetapher, für die die Funktionsobjekte eine spezifische Interpretation sind, erneut in veränderter Form aufgenommen wird (cf. Abschnitt 4.4), soll erst ein anderes, bei graphisch-direktmanipulativen Systemen beobachtbares Standardverfahren diskutiert und generalisiert werden, die Verwendung von Formularen.

4.3 Formularmodus und Navigationsleitlinien

Bevor weiter nach einer Art der Objektorientierung gesucht wird, die die geschilderten Nachteile vermeidet, soll nochmals ein Ansatz aus der Diskussion von N/JOY aufgegriffen werden, der ebenfalls die Handlungsorientierung zurückzudrängen in der Lage ist, die spezielle Ausprägung des Dialogboxkonzepts.

4.3.1 Formulare als Erklärungsmetapher für eine spezielle Form der Objektorientierung

Die Objekte von N/JOY haben vordefinierte Eigenschaften, die sich bei Bedarf über eine Dialogbox verändern lassen. In unserem Zusammenhang ist wesentlich, daß die Einträge in den Dialogboxen stets Attribuierungen, keine Handlungsanweisungen sind. Über die OK-Taste werden die Attribuierungen dem Objekt zugewiesen.

Auch bei traditionellen graphisch-direktmanipulativen Systemen treten diese OK-Dialogboxen auf. Sie stehen neben einer handlungsorientierten Sichtweise in den Dialogboxen, die sich durch die Existenz von Aktionstasten mit direkten Handlungsanweisungen definieren lassen.

OK-Dialogboxen können metaphorisch als Formulare angesehen werden, deren Überschriften eine Handlung (z. B. Suche in der Datenbank) bezeichnen. Im Formular selbst gibt der Benutzer nur Parameter an, die diese Handlung spezifizieren. Erst wenn das Formular fertig ausgefüllt ist (=OK), wird es in der realen Welt an eine handelnde Institution weitergegeben, die aktiv wird.

Deutlichstes Zeichen für diese Art der Konzeptualisierung ist die OK-Taste, mit der die getroffenen Festlegungen bestätigt werden.

Im Prinzip sind OK-Dialogboxen, die hier mit Hilfe der Formularmetapher erklärt wurden, nichts anderes als weiterentwickelte Bildschirmmasken der Vor-Windows-Zeit, die vor allem für Datenbankabfragen beliebt waren (und noch sind). Bei all ihrer Unflexibilität - die es letztlich verbietet, sie als alleiniges Gestaltungsmittel einzusetzen, wie dies bei DOS-Systemen oft geschah - haben vorgefertigte Eingabemasken für Datenbankabfragen einen deutlichen Vorteil. Der Benutzer benötigt keinerlei Kenntnisse von Datenbankstrukturen. Abfragen dieser Art sind denkbar einfach zu formulieren.

In diesen Zusammenhang gehört auch eine andere frühe Gestaltungsform von Datenbankabfragen: Query by example. Dem Benutzer werden die ausgewählten Tabellen der Datenbank graphisch am Bildschirm vorgelegt. Er trägt in die betreffenden Felder Werte und Restriktionen ein, die die Ergebnisauswahl steuern. Query by example läßt sich somit auf das Ausfüllen von Tabellen zurückführen. Damit wird ein Großteil der SQL-Syntax, der die Zuordnung von Werten und Bedingungen zu den Tabellen regelt, überflüssig. Solange keine zweite Tabelle ins Spiel kommt, entspricht dieser Modus dem des Ausfüllens eines Formulars. Sobald bei der Formulierung der Suchanfrage jedoch die Join-Operation notwendig wird, gilt das gleiche wie für alle kommandoorientierte Verfahren (hier SQL); der Benutzer muß EDV-motiviertes Wissen einsetzen. Befragungen im Rahmen des Projekts WING-IIR zeigten, daß Benutzer diesen Bruch deutlich als störend empfinden und Fehler machen, im Gegensatz zum Grundeinstieg über die graphische Darstellung der Tabellenstruktur.

Es gibt somit eine Reihe guter Gründe, von Formularen auszugehen, um eine einfache, möglichst wenig handlungsorientierte Basis besonders für Anfänger zu bekommen. Deshalb wurde der Formularmodus in das WOB-Modell integriert.

„Formulare" in diesem Sinn sind allerdings nicht die der Papier-und-Bleistift-Welt. Sie enthalten im WOB-Modell Gestaltungselemente wie Klapplisten oder Auswahlboxen. Entscheidend ist, daß man mit Formularen nichts „tut". Man spezifiziert ein Tun, das später, nachdem alle Angaben gemacht sind, durch einen Dritten ausgeführt wird, der das Formular erhält. Diese Grundkonzeptualisierung, bei der die Handlung selbst in den Hintergrund tritt, soll vermittelt werden. Sie ist ein entscheidendes Gestaltungselement für den WOB-Ansatz.

Die Verwendung des Formularbegriffs in diesem erweiterten Sinn läßt sich auf zwei Wegen rechtfertigen:

- als „natürliche" Extension des Formularbegriffs in die elektronische Welt hinein, vergleichbar mit der - vom Benutzer als selbstverständliche Erweiterung empfundenen - Eigenschaft von Papierkörben innerhalb der Desktopmetapher, Schrankikonen aufnehmen zu können,
- als „moderne" umgangsprachliche Sprachverwendung, die sich durch elektronische Formularprogramme weiterentwickelt hat. Interessant ist z. B., daß Formulare in Word für Windows Version 6 ebenfalls Klapplisten und Auswahllboxen enthalten können, ohne daß deshalb ein anderer Begriff für das elektronische Ausfüllen und Verwalten von Formularen geschaffen würde. Geht man davon aus, daß die elektronischen Formulare der Textverarbeitung die Semantik des Begriffs „Formulars" schon weitgehend verändert haben, braucht nicht mehr mit dem Erklärungsmechanismus der „natürlichen" metaphorischen Erweiterung gearbeitet werden.

Ein weiterer Vorteil der Verwendung von Fomularen ist, daß sich Navigationsleitlinien einfach als Folge sich selbständig öffnender Formularfenster abbilden lassen, deren Abfolge kognitive Standardsituationen der Suche darstellen. So sind in WING-M2 die drei Varianten der Suchfenster die zentralen Eröffnungselemente, die einerseits nach den beiden Suchmodalitäten und andererseits nach dem Kriterium spezifische Information vs. Überblicksinformation differenziert sind. Weitere Fenster (z. B. zur weiteren Präzisierung durch Temperaturangaben) öffnen und schließen sich, je nach dem Fortgang der Interaktion. D. h., automatisch wirkende Kontrollstrukturen entlasten den Benutzer so weit, daß sich die Systemvorstellung im wesentlichen auf das Ausfüllen von Formularen reduzieren läßt, ohne zu Fehlbedienungen zu führen.

Interessanterweise machte die Datenbankabfrage mit dem Suchwerkzeug „Formular" bei den WING-Tests zwar keine Schwierigkeiten, die Benutzer sprachen selbst jedoch nur selten von Formularen, meistens von Bildschirmen oder Masken. Als sich dies bei Voruntersuchungen andeutete, wurden die Versuchsleiter nochmals explizit angewiesen, den Term bei der Kurzeinführung deutlich zu benutzen, ohne daß dies Auswirkungen gehabt hätte.

Das Ergebnis irritiert, weil man erst einmal annehmen müßte, daß eine nicht akzeptierte Metaphorik zu Bedienschwierigkeiten führt. Der Grund liegt wohl darin, daß über den Metaphernbegriff im Prinzip nur die Attribuierungsstrategie transportiert wird. Der Benutzer sollte mit Hilfe der Metapher die objektorientierte Sichweise einnehmen, die die OK-Taste ausdrückt. Of-

fensichtlich braucht er dazu keine Metapher, zumindest nicht bei der Systembedienung. Die Formularmetapher funktioniert als Erklärungsmetapher, die die Philosophie von OK-Dialogboxen transportiert. Einmal verstanden, scheint die Metapher aber vielen Benutzern semantisch zu schwach, um den Aufwand zu rechtfertigen, sich die objektorientierte Grundphilosophie indirekt über den metaphorischen Analogieschluß zu merken. Der Benutzer wendet die „Formular"- Suchobjekte direkter an, im Sinne von visual formalisms, mit denen sich Abschnitt 5 auseinandersetzt.

Insofern könnte man im WOB-Modell auf den Formularbegriff ganz verzichten. Als sprachliche Erklärungsmetapher wird der Formularbegriff jedoch weiter benutzt, aus historischen Gründen und der Einfachheit der Ausdrucksweise wegen. Es spricht auch nichts dagegen, bei der Ersteinführung der WOB-Dialogboxen die objektorientierte Sichtweise über den Vergleich mit den Formularen des täglichen Lebens nahe zu bringen. Daß die Formulare im WOB-Modell als „visual formalisms", nicht mehr als Systemmetapher wirken, soll die Bezeichnung Formularmodus ausdrücken.

4.3.2 Mangel an Flexibilität

Das vorgeschlagene Formularkonzept zur Datenbankabfrage enthält allerdings - für sich genommen - noch zu wenig Flexibilität. Sie liegt höchstens in der Erzeugung und Spezifizierung der Formulare durch einen Administrator, der die Bedürfnisse seiner von ihm betreuten Benutzergruppe kennt.

Benutzergruppen, die durch einen Administrator bedient werden, dürften jedoch in den meisten Fällen immer noch zu einem nicht unbeträchtlichen Anteil inhomogen sein.

Zudem ist eine Hauptgruppe flexibler Systemanpassung an individuelle Benutzer und Nutzungssituationen bisher nicht möglich: die Voreinstellung bestimmter Parameter. Sucht der Benutzer z. B. immer nur nach einem bestimmten Kundentyp oder scheiden bestimmte Versicherungsformen in einem Unternehmen von vornherein aus, sollte eine einfach bedienbare Vorbelegung möglich sein.

Als zentral hat sich somit für die weitere Ausgestaltung des WOB-Modells die Überlegung herausgestellt, - unter einer einheitlichem Modellvorstellung - einen Ansatz zu finden, der den flexiblen Übergang zwischen verschiedenen Benutzergruppen und Nutzungssituationen gewährleistet.

4.4 Zustandsanzeige mit Korrekturmodus

In diesem Abschnitt wird vor allem eine Lösung für die dynamische Rücknahme der Selbsterklärungsfähigkeit eines Systems gesucht (cf. Abschnitt 3.2.1). Die hierfür geforderte Flexibilität muß das Kontinuum zwischen dem Erstbenutzer von Software ohne EDV-Vorkenntnisse und dem routinierten Benutzer abdecken, der ein System seit längerer Zeit regelmäßig bedient.

4.4.1 Komprimierter Eingangsbildschirm

Die folgenden Überlegungen werden anhand eines stark vereinfachten Reise- und Hotelauskunftsystems verdeutlicht, das keinen Anspruch auf Anwendungsadäquatheit erhebt. Es wurde so konstruiert, daß die Grundidee der Flexibilisierung durch das Gestaltungsmittel komprimierter Eingangsbildschirme möglichst deutlich hervortritt.

Bevor Lösungsvorschläge auf dieser Basis folgen, soll der Zusammenhang zwischen Zustandsanzeige und Ergebnisdarstellung präzisiert werden.

4.4.1.1 Zustandsanzeige und Ergebnisdarstellung

Ein Großteil der Suchprozesse am Computer verläuft iterativ. Es steht z. B. nicht von Beginn an fest, ob ergänzende Parameter wie die Forderung nach einem Faxgerät im Hotelzimmer sinnvoll sind (eventuell zu wenige Treffer). Deshalb setzt sich ein Suchprozeß in der Regel aus mehreren Teilschritten zusammen, bei denen der Benutzer ursprünglich gewählte Parameter ändert. D. h., er setzt eine mehrstufige Problemlösungsstrategie ein, in deren Verlauf sich die ursprüngliche Problemlage wandelt (cf. Ingwersen 1994, Bates 1989).

Suchprozesse dieser Art sind in hohem Maße davon abhängig, daß der Benutzer bei jedem Teilschritt weiß, welche von ihm gewählten Parameter zur vorliegenden Ergebnismeldung führten. Dies trifft prinzipiell auch auf mehrstufige, zielgerichtete Suchprozesse zu, obwohl hier die Chance, daß sich der Benutzer seine vorangegangenen Eingaben merkt, größer ist.

Zustandsanzeigen kosten jedoch Platz, genauso wie die Selbsterklärungsfähigkeit.

Andererseits verlangt die Ergebnisauswertung oft, daß möglichst viele der gefundenen Ergebnissätze gleichzeitig am Bildschirm erscheinen (z. B. alle Hotels einer Sparte in einem Ort). Daß dieser Bereich wesentlich kleiner als

beim Druckmedium ist, behindert den Benutzer, der sich aus der Ergebnisliste mit Hilfe eigener Bewertungsverfahren einige wenige auswählen möchte.

Deshalb ist die Größe der Darstellungsfläche, die die Ergebnisliste einnehmen kann, in vielen Anwendungssituationen ein wichtiges Kriterium. Es steht in einem natürlichen Gegensatz zu der Forderung nach einer möglichst vollständigen Zustandsanzeige und zur Selbsterklärungsfähigkeit. Deren Verbesserung muß in der Regel entweder mit dem Verbrauch von Bildschirmplatz für die Ergebnisdarstellung oder mit längeren Bedienwegen erkauft werden.

4.4.1.2 Lösungsidee

Vorgeschlagen wird ein System sich gegenseitig überlagernder Fenster, die sich als auszufüllende Formulare interpretieren lassen (cf. Abschnitt 4.3). Die Größe der Fenster und ihre Position am Bildschirm ist standardmäßig vorgegeben und orientiert sich an einem Benutzer ohne Vorkenntnisse. Bei steigender Vertrautheit mit dem System kann die Selbsterklärungsfähigkeit zurückgenommen werden. In unserem Beispiel weiß der Benutzer z. B. binnen kurzem, daß er nach der Kategorie „Sonderservice“ differenzieren kann und welche Einträge diese Auswahlbox hat. Deshalb lassen sich redundante Elemente ausschalten bzw. überdecken. Dies ist so zu konstruieren, daß sich die Bedienwege verkürzen und/oder die Bildschirmfläche für die Ergebnisdarstellung vergrößert wird. D. h., der Benutzer hat die Möglichkeit, punktuell und dynamisch nicht mehr benötigte Selbsterklärungsfähigkeit gegen Platz für die Ergebnisdarstellung oder gegen Zeit (weniger Bedienschritte) zu tauschen.

Wesentliches Element des Designentwurfs ist die Überlegung, daß das während des iterativen Aufbaus einer Suchfrage sukzessiv anwachsende Merkpotential genutzt werden sollte. So läßt sich das Hauptsuchformular, mit dem der Anfänger eine neue Suchanfrage beginnt, strikt selbsterklärend auslegen. Für Änderungen und Erweiterungen der Suchanfrage kann der Benutzer auf eine andere, komprimiertere Darstellungsform umgeschalten, da er die Möglichkeiten des Hauptbildschirms noch im Kopf hat. Das Bedürfnis nach einer Zustandsanzeige und der Wunsch, Änderungen der Suchanfrage ohne Bildschirmwechsel vornehmen zu können, werden zusammengeführt. Mit dem Ergebnisbildschirm erscheint in komprimierter Form eine Darstellung der im vorangegangenen Hauptbildschirm ausgewählten Parameter, die gleichzeitig als Zustandsanzeige und als Eingabefläche für Änderungen dient (cf. Abschnitt 4.4.1.3).

Die übliche, rein darstellende Zustandsanzeige nutzt dagegen die Bildschirmfläche nur für diesen einen Zweck. Wählt man andererseits ein Verfahren, bei dem das Suchformular maximal 1/3 der Bildschirmfläche ausfüllen darf und permanent am Bildschirm verbleibt, müßte in vielen Fällen ein Mittelweg zwischen Selbsterklärungsfähigkeit und Platzeinsparung zugunsten der Ergebnis-Bildschirmfläche gefunden werden, auch dort, wo die Menge der Suchparameter dies nicht mehr befriedigend zuläßt.

Interessant ist, daß das Betriebssystem WINDOWS95 mit seiner START-Leiste am unteren Bildschirmrand die Zustandsanzeige ebenfalls durch Manipulationsmöglichkeiten anreichert. Der Benutzer kann zwar keine Änderungen durchführen, durch Anklicken jedoch alle hier angezeigten Fenster öffnen und in den Vordergrund stellen.

(Daß der Benutzer, der WINDOWS95 *beenden* will, dieses Kommando unter dem Oberbegriff *Start* findet, gehört wohl zur heute leider „normalen“ Vorstellung einer softwarergonomischen „Optimierung“ in den firmeneigenen „usability labs“.)

Natürlich wird es immer wieder Suchformulare geben, bei denen eine sinnvolle Komprimierung nicht möglich ist. In diesem Fall entfällt eine WOB - Option, die jedoch bei einer großen Gruppe von Anwendungsfällen wesentlich und hilfreich sein dürfte.

Die Hauptelemente der bisher formularorientiert möglichen Datenbankabfragen sind somit:

a) Das Hauptsuchformular (Hauptsuchbildschirm), das - nach Vorgabe der vom Benutzer anzugebenden Parameter - die Suche in der zugrundeliegenden Datenbank spezifiziert.

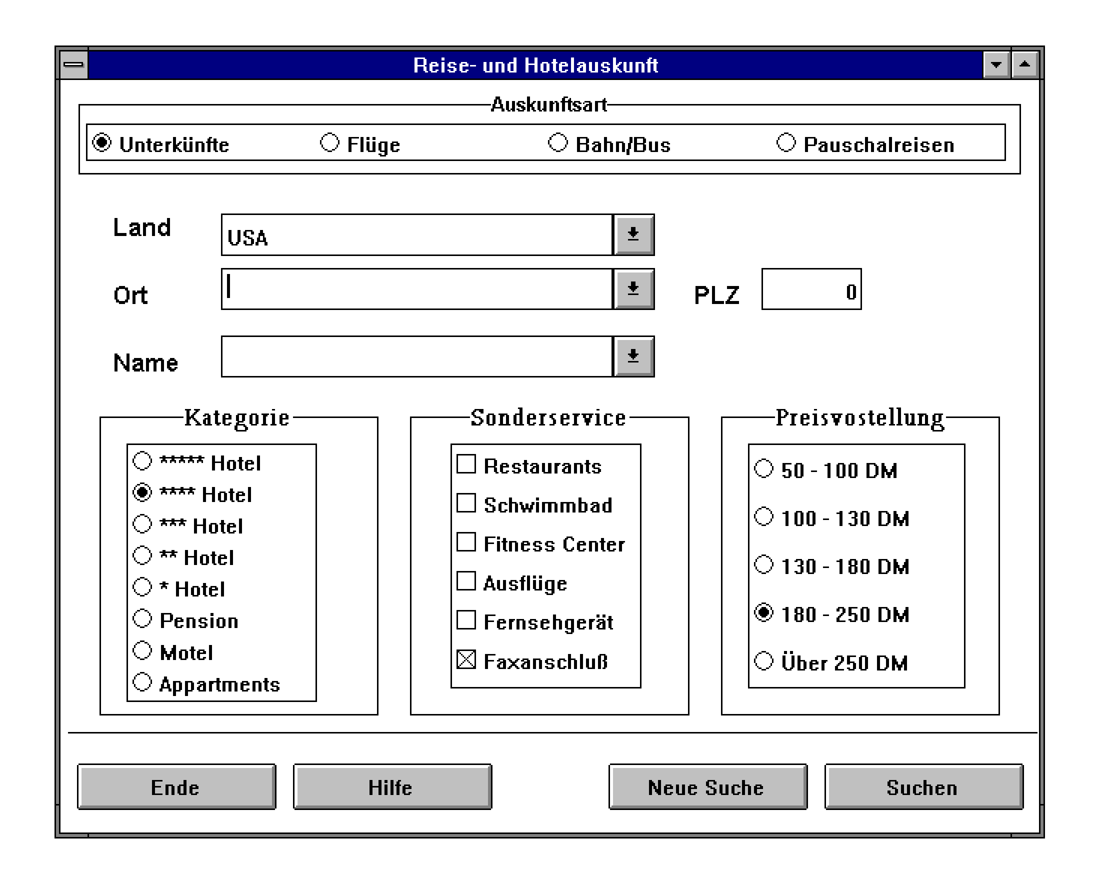
b) Das komprimierte Suchformular in der Form des Suchparameterfensters, das nur eine andere Oberflächenform von a) sowie gleichzeitig Zustandsanzeige ist und Änderungen ermöglicht (oberer Bereich des Suchparameter- und Ergebnisbildschirms).

c) Das Ergebnisformular, dessen Fenster in Kombination mit dem Suchparameterfenster auftritt und sozusagen ein „Loch“ im Suchparameterfenster füllt. Es ist Ergebnis der Anwendung des Suchformulars auf die Datenbank (unterer Bereich des Suchparameter- und Ergebnisbildschirms).

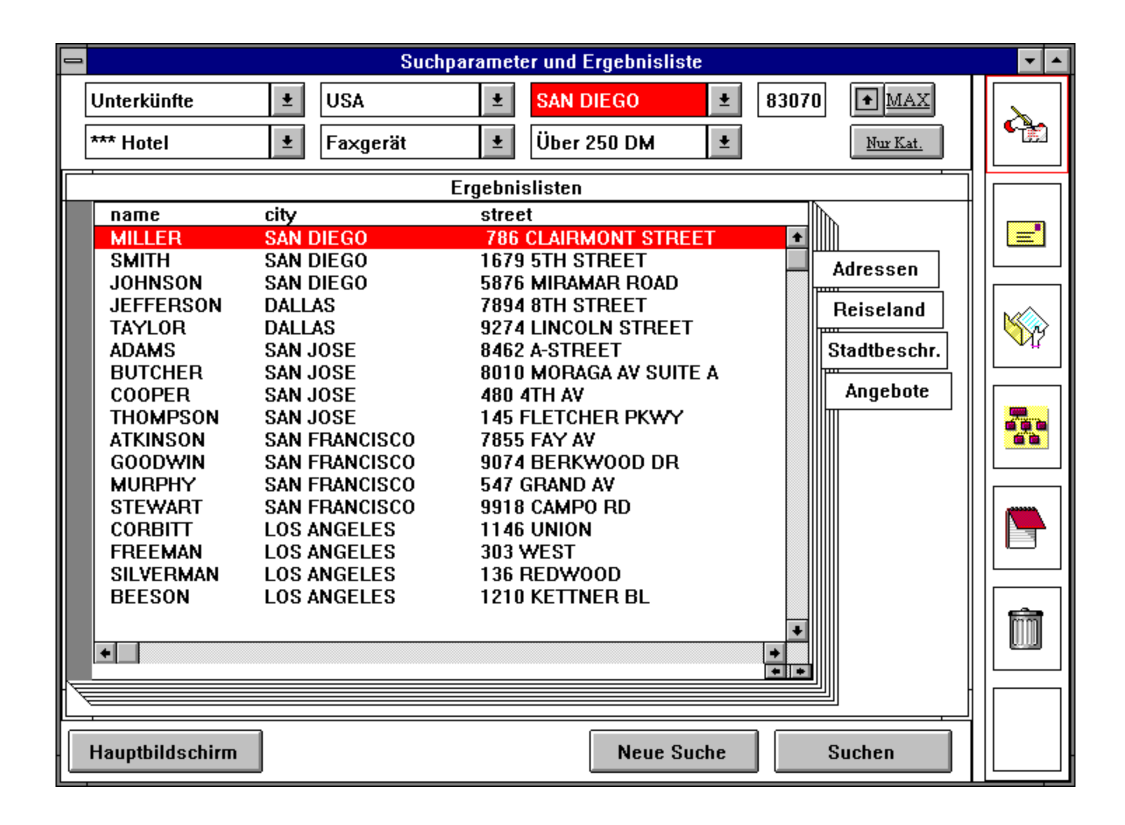
4.4.1.3 Bildschirmgestaltung und Präzisierung am Beispiel ROTEL

Im folgenden wird die in Abschnitt 4.4.1.2 abstrakt skizzierte Lösungsidee in detaillierte Gestaltungsvorgaben für das reduzierte WOB-Beispiel ROTEL umgesetzt und der flexible Austausch von Selbsterklärungsfähigkeit gegen Platz für die Ergebnisdarstellung durch zusätzliche Techniken verfeinert. Entscheidend ist dabei, eine Form zu finden, die einen möglichst stufenlosen Übergang ermöglicht.

Die Bildschirmskizzen beziehen sich auf den Zustand, den der Anfangsbenutzer wahrnimmt, der weder weitreichende Erfahrung mit Windows-Applikationen hat, noch die Anwendung von früheren Benutzungen her kennt.



ROTEL Hauptsuchbildschirm (Ebene 1)



ROTEL Bildschirm Ebene 2

Für Benutzer, die mit der Bedienung von ROTEL bereits vertrauter sind, kann der Hauptsuchbildschirm unterdrückt werden. In diesem Fall wird die Suchanfrage ausschließlich in der komprimierten Form des Suchparameterfensters eingegeben.

Die Gestaltung des ROTEL Suchparameter- und Ergebnisbildschirms enthält einige Detailentscheidungen:

- Die erste Auswahlgruppe des Hauptbildschirms führt zur einer dynamischen Anpassung des unteren Bildschirmbereichs (cf. Abschnitt 4.1).

  Eine andere Form der Umsetzung wären Referenzkarten unter Windows oder das Notizbuchkonzept unter OS/2.

- Die Funktionen 'ENDE' und 'HILFE' sind am besten in der Überschriftszeile aufgehoben, da sie einen Sonderstatus haben (wie z. B. auch in GEOWORKS; cf. auch die Erweiterungen in der Titelleiste von WINDOWS95). Da Standard-GUI-Tools derzeit noch nicht zulassen, die Überschriftszeile für spezielle Aktionstasten zu nutzen, wird als Alternative die

Positionierung von ‘ENDE’ und ‘HILFE’ in der unteren Tastenzeile vorgesehen.

4.4.1.4 Flexibilität durch Fenstervariation

Die obige Abbildung des Suchparameter und Ergebnisbildschirms von ROTEL enthält programmtechnisch zwei sich überlagernde Fenster, die sich gleichzeitig paßgenau übereinander öffnen. Der obere Bereich ist nichts anderes als eine andere, komprimierte, weniger selbsterklärende Darstellung des Hauptsuchformulars (Erst-einstieg). Für den Anfänger wirken beide Fenster zusammen als einheitlicher Ergebnisbildschirm.

Ein weiterer Vorteil des vorgeschlagenen Modells liegt darin, daß Benutzer das Ergebnisfenster standardmäßig und in beliebiger Form vergrößern können. Hat der Benutzer z. B. die Funktionstasten der letzten Zeile des Suchfensters im Kopf, kann er sie durch eine Erweiterung des Ergebnisfensters überdecken und beide Funktionen über die Tastatur auslösen. Arbeitet er immer im gleichen Verzeichnis (oder auch nur für eine längere Zeit) und im gleichen Suchmodus, sollte er das Ergebnisfenster zudem nach oben erweitern können (mit Maus ziehen). Dabei korrigiert das System die vom Benutzer gesetzte obere Grenze immer so, daß nur vollständige Elemente unüberdeckt bleiben.

Beim Verbandsinformationssystem ELVIRA (cf. Abschnitt 4.1), bei dem sich das Ergebnisfenster über den Browserbereich des Eingangsbildschirms legt, führt eine Vergrößerung des Ergebnisfensters nach oben dazu, daß die Eingabefelder der drei Facetten mit einer scrollbar versehen werden, sobald ein Deskriptor verdeckt wird. Diese Lösung bietet sich immer dann an, wenn die Zustandsanzeigen Listeneinträge enthalten, d. h. Mehrfachauswahl zugelassen ist.

Eine weitere Besonderheit der Anwendung des Prinzips des komprimierten Eingangsbildschirm als Zustandsanzeige in ELVIRA ist, daß die Felder bereits beim Eingangsbildschirm die Funktion der Zustandsanzeige für einzelne Schritte der Browserauswahl haben - neben der Direkteingabe von Termen.

Programmierproblem beim Einsatz von Standard GUI-Tools: Die flexible Erweiterung nach oben setzt voraus, daß beim Fokussierungswechsel zum Suchparameterfenster die Texteingabefelder nicht wieder erscheinen (aktives Fenster nach vorn). Deshalb kann es in der Praxis notwendig werden, die Flexibilität auf die folgenden Zustände der Höhenveränderung des Ergebnisfensters einzuschränken:

a) Standardform, die sowohl die untere Tasten-Leiste als auch das obere Suchparameterfenster freiläßt.

b) Erweitern des Ergebnisfensters nach unten, so daß die Tastenleiste überdeckt wird. Zurück zum 'HAUPTBILDSCHIRM', 'Neue Suche' und 'SUCHEN' werden dann vom Benutzer ausschließlich über Funktionstasten ausgelöst.

c) Ergebnisbildschirm als Vollbildschirm

Diese Form läßt wiederum zwei Bedienungsvarianten zu:

c1) Expertenmodus:

Bedienung vollständig über die Funktionstasten der Tastatur bei ausgeschaltetem Hauptbildschirm und ohne Zustandsanzeige.

Dies würde voraussetzen, daß die Funktionstastenbelegung um den Modus eines sich überlagernd öffnenden Eingabefensters für Ort, Name etc. erweitert wird, das sich bei der Eingabe von ENTER sofort wieder schließt.

Zumindest ein Teil der Zustandsanzeige könnte in diesem Fall in die Überschriftszeile integriert werden.

c2) Maximales Ergebnisfenster und Hauptbildschirm zur Formulierung der Anfragen

Der Modus ist für Anfänger geeignet, die auf eine maximal erweiterte Ergebnisliste (ganzer Bildschirm ohne Suchparameterbereich) nicht verzichten möchten, die Funktionstastenbedienung jedoch noch nicht sicher gelernt haben. Der Preis wäre der doppelte Bedienschritt. Bei jeder Änderung und neuen Suche schaltet der Benutzer auf den Hauptbildschirm zurück und setzt dort seine Parameter.

Alternativ ließe bei dieser Lösung die Vergrößerung des Ergebnisbildschirms die untere Buttonleiste frei, womit auch bei den Standardfunktionen auf eine Merkleistung des Benutzers verzichtet werden könnte.

Hat ein Benutzer alle Kommandos als Funktionstastenaufrufe im Kopf, schaltet er den Hauptbildschirm ab und expandiert das Ergebnisfenster über den ganzen Bildschirm (c1). Damit ist das Optimum an Platzausnutzung und Reduzierung der Bedienschritte erreicht. Kennt er nur einen Teil der Funktionsaufrufe und will trotzdem nicht auf die maximale Ausdehnung des Ergebnisfensters verzichten, läßt er den Hauptbildschirm

eingeschaltet (c2) und greift immer nur dann auf ihn zurück, wenn er sich nicht an einzelne Funktionstastenbelegungen erinnern kann.

Der Benutzer kann die Selbsterklärungsfähigkeit des Systems somit flexibel zurücknehmen, um mehr Platz für die Ergebnisdarstellung zu gewinnen, sobald er die Unterstützung nicht mehr notwendig hat.

Das Verfahren läßt sich bei größeren Bildschirmen (z. B. 19 Zoll) zudem so nutzen, daß die Fenster nebeneinander gelegt werden (dann ohne komprimiertes Suchparameterfenster).

4.4.1.5 Suchparameter- und Ergebnisbildschirm bei abgeschaltetem Hauptsuchbildschirm

Das optionale Abschalten des Hauptsuchbildschirms überführt die Anwendung in ein Ein-Bildschirm System, das die Selbsterklärungsfähigkeit in der komprimierten Form dauerhaft zurücknimmt. Intern existiert weiter der view 'Hauptformular', auf dem die nun allein sichtbar bleibende komprimierte Form des Suchformulars basiert.

4.4.1.6 Einschub: Metaaktionen

Die ROTEL-Beispiele enthalten mit der Tastenaufschrift „Suchen“ einen Verstoß gegen die Gestaltung der Tasten im Formularmodus. Nach Abschnitt 3.2.4.2 und Abschnitt 4.3 sind auf den Tasten keine Aktionsbezeichnungen erlaubt. Diese Regelung führt dazu, daß im WOB-Modell versucht wird, auch bei Aktionen der Metaebene wie „Beenden, Suchen, Löschen Feldinhalte, Abbrechen“ Terme zu finden, die sich als statische Parametersetzungen interpretieren lassen, z. B. „Ende, Abbruch“ usw. Gegen diese Festlegung gab es immer wieder vereinzelt Widerstände von Benutzern und Entwicklern. Beispielimplementationen, die sich auf der Ebene der Metaaktionen (! aber nur hier) über die WOB-Regel hinwegsetzten, konnten nie empirisch als fehleranfälliger oder schlechter zu vermitteln nachgewiesen werden.

Eventuell handelt es sich um eine Übergeneralisierung der Ergebnisse von WING-M1. Möglicherweise läßt die Ebene der Metaanweisungen, die z. B. nicht die Festlegung der Suchbedingungen betreffen, sondern die oberste Steuerebene der Software darstellen, durchaus Handlungsanweisungen zu. In diesem Fall würde der Benutzer beim Aufbau seines mentalen Modells zwischen der direkten Handlungsebene des Programms und der Metaebene, bei der die Programmelemente selbst Objekt der Benutzermanipulation sind,

problemlos unterscheiden. Warum das so ist (bzw. ob dies wirklich so ist), darüber läßt sich derzeit nur spekulieren. Vielleicht liegt es einfach daran, daß die hier in Frage kommenden Anweisungen bereits so häufig in graphischen Oberflächen auftreten, daß sie als Altwissen gelten können, die einzeln - ohne den Hintergrund eines mentalen Modells - abrufbar sind. Vielleicht muß bei der Modellbildung die Metaebene auch deutlich gesondert konzeptualisiert werden.

Die Frage, wie bei diesem Problem in der Praxis zu verfahren ist, sollte derzeit am besten individuell, anwendungsbezogen und empirisch entschieden werden.

Trotzdem ist Vorsicht geboten. Kritisch bleibt bei einer halbverstandenen Mischung von Tasten mit und ohne Handlungsbezeichnungen die mögliche Destablilisierung der Sichtweise einer strikten Objektorientierung, die dem Benutzer möglichst in jedem Detail ohne inhärente Widersprüchlichkeit signalisiert werden soll. Insofern ist man generell auf der sichereren Seite, wenn man so weit wie möglich ohne Aktionsbezeichnungen auf Tasten auskommt. Im Bereich der Metaaktionen läßt sich z. B. der Navigationskomplex prinzipiell besser durch das Muster „Pfeil + Zielangabe“ ausdrücken (z. B. statt: Gehe zu ...“), das auf die direkte Raumerfahrung des Menschen rekurriert (cf. Krause 1996).

4.4.1.7 Fazit

Die Technik, Flexibilität durch Rücknahme der Selbsterklärungsfähigkeit mit Hilfe des komprimierten Eingangsbildschirms und der beschriebenen dynamischen Überlagerungstechniken zu erreichen, wird immer dann scheitern, wenn die Anwendung zu komplex wird. Dann müßte die notwendige Selbsterklärungsfähigkeit auf eine Kaskade hintereinandergeschalteter Bildschirme verteilt werden. Dieser Fall ist z. B. bei WING-Petten gegeben (cf. Roppel 1996) und eingeschränkt auch bei WING-M2.

Anhand von WING-M2 soll im folgenden Abschnitt eine Alternative der Flexibilitätserhöhung besprochen werden. Sie basiert wiederum auf dem Grundmuster „Zustandsanzeige mit gleichzeitiger Korrekturmöglichkeit“, nutzt jedoch eine Modalitätsmischung zwischen natürlicher Sprache und graphisch-direktmanipulativem Modus.

Roppel 1996 sucht dagegen eine Lösung durch die Integration einer verstärkten Visualisierung. Er arbeitet für die Werkstoffparameter mit graphisch

dargestellten hierarchischen Strukturbäumen, deren Umfang durch eine Benutzermodellierungskomponente reduziert wird.

4.4.2 Natürlichsprachliche Zustandsanzeige über Generierungsalgorithmen und Korrekturmodus

Wie in Abschnitt 3.2.1 diskutiert, führte die Modalitätsmischung von natürlichsprachlicher Abfragemöglichkeit und graphisch-direktmanipulativem Verfahren in WING-M1 nicht zum erwarteten Ergebnis. Die Testpersonen der MTU nahmen den parallel angebotenen natürlichsprachlichen Modus nicht an, obwohl er bei den Tests mit dem simulierten natürlichsprachlichen Interface als alleinigem Zugang durchaus positiv bewertet wurde.

Die Ausgangsthese für die Gestaltung von WING-M2 auf der Basis der Auswertungen der Protokolle war: Die geringe Nutzung der natürlichsprachlichen Modalität könnte darin begründet sein, daß der Designentwurf von WING-M1 die speziellen Vorzüge der Natürlichsprachlichkeit gegenüber dem graphischen Modus nicht genügend stark unterstützt.

Somit war - unter dem Aspekt der Modalitätsmischung - nach einer Lösung zu suchen, die die Einsatzmöglichkeiten der natürlichsprachlichen Modalität ausweitet und die Motivation der Benutzer, sich ihrer zu bedienen, erhöht.

Wo liegen die Hauptvorteile des graphisch-direktmanipulativen Modus?

Die Benutzer von WING-M1 bevorzugten ihn vor allem wegen des ihm inhärenten Vorlagecharakters: Alle Alternativen sind explizit angeboten und müssen nur noch passiv erkannt und mit der Maus ausgewählt werden. Nachteilig hieran ist, daß sich die Informationen wegen des großen Platzbedarfs auf mehrere Bildschirme verteilen, weshalb der Benutzer die Übersicht über vorher getroffenen Entscheidungen verlieren kann.

Der erste Schritt, diesen Nachteil auszugleichen, ist die Einführung einer expliziten Zustandsanzeige aller getroffenen Parametersetzungen. Die HTM-Datenbank von Petten (cf. Roppel 1996) bedient sich hierzu z. B. einer vereinfachten SQL-Notation, die sich als quasi-natürlichsprachlicher Formalismus auch ohne SQL-Kenntnisse interpretieren läßt.

Das WOB-Modell verbindet die Forderung nach einer Zustandsanzeige in WING-M2 mit einer Ausweitung der natürlichsprachlichen Eingabemöglichkeiten. Als Zustandsanzeige wird eine natürlichsprachliche Formulierung benutzt, die das System schrittweise aus den graphischen Eingaben generiert

und die in einem eigenen Fenster unterhalb des Fensters für die graphische Eingabe erscheint. Die neue Funktionalität verbindet sich somit mit der weiter bestehenden Möglichkeit, rein natürlichsprachliche Suchanfragen zu stellen. Beide Fenstertypen, der graphisch-direktmanipulative und der natürlichsprachliche, sind nur verschiedene Ausdrucksweisen der gleichen Suchanfrage. Sie werden kontextsensitiv und dynamisch in beiden Richtungen miteinander verbunden. Zu jeder im Graphikmodus vom Benutzer festgelegten Parametrierung generiert das System automatisch einen natürlichsprachlichen Ausdruck (und umgekehrt).

Die entscheidende Verbesserung des Einsatzes der natürlichsprachlichen Modalität besteht darin, daß in der vom System generierten natürlichsprachlichen Formulierung korrigiert werden kann. Will der Benutzer z. B. bei einer komplexen Anfrage nur nachträglich die Temperaturangabe erhöhen, nutzt die dem graphischen Modus inhärente Vorlageleistung nichts. Es bleibt nur der Nachteil, daß das entsprechende Teilfenster extra angesteuert werden muß. Der natürlichsprachliche Korrekturmodus ist in diesem Fall überlegen.

Gleichzeitig erlaubt das Verfahren dem Benutzer, auch innerhalb einer Anfrage beliebig zwischen beiden Modi zu wechseln. Bedingungen, für die kein Vorlagecharakter notwendig ist, lassen sich jetzt ohne Umwege natürlichsprachlich formulieren. Der Benutzer kann an jeder Stelle der Interaktion sofort wieder in den graphischen Modus wechseln (und umgekehrt), wenn er sich Vorteile davon verspricht.

Diese mehrfache Nutzung der natürlichsprachlichen Modalität als alternativer Eingabemodus, als Korrekturmodus und Zustandsanzeige erwies sich bei den WING-M2 Tests als erfolgversprechend (cf. Marx 1995). In dieser Form wurde die Natürlichsprachlichkeit angenommen und gleichzeitig die in WING-M1 bestehende Lücke der fehlenden Zustandsanzeige geschlossen.

Prinzipiell liegt dieser Lösung, die im WOB-Modell als Alternative zur Technik der komprimierten Eingangsbildschirme verwendet wird, die gleiche Grundstrategie zugrunde. Der komprimierte Eingangsbildschirm aus Abschnitt 4.4.1 besteht aus nichts anderem als den sinntragenden Wörtern des natürlichsprachlichen Modus, allerdings ohne die syntaktische Komponente. Die Plazierung in Datenfeldern kostet zusätzlichen Platz, erlaubt aber gleichzeitig die Klapplisten, die den reduzierten Vorlagecharakter auf einer nachgeschalteten Ebene punktuell wieder einfügen können.

Das WOB-Modell läßt sich allerdings schon deshalb nicht ausschließlich auf die Technik der Modalitätsmischung gründen, weil der natürlichsprachliche Generierungsalgorithmus für jede einzelne Applikation zumindest vom Wortschatz her anzupassen ist. Auch wenn es gelingt, diesen Aufwand auf wenige Wochen zu reduzieren, dürften Benutzer dies für viele Applikationen nicht akzeptieren, weshalb auf jeden Fall eine Alternative vorhanden sein muß.

Die korrigierbare, natürlichsprachliche Zustandsanzeige wirft eine Reihe von theoretischen und praktischen Umsetzungsproblemen auf, die sich nicht mehr durch eine einfache Adaption bestehender Grammatikmodelle und Parsingstrategien aus der Computerlinguistik lösen lassen. Ihnen geht Marx 1995 bei der Entwicklung der Komponente WING-NL für das Werkstoffinformationssystem der MTU nach. In der für das WOB-Modell notwendigen Form wurde die Sprachgenerierung und -analyse bisher von der Computerlinguistik nicht behandelt, einfach deshalb, weil es sich um ein neuartiges Prinzip der Einbindung von natürlichsprachlichen Teilkomponenten in ein Fakteninformationssystem handelt:

a) Es geht bei der Generierung nicht um SQL, sondern vorrangig um eine Generierung aus einer graphischen BOF. Die zu generierenden natürlichsprachlichen Ausdrücke beziehen ihre Semantik somit aus controls wie Eingabefeldern oder Auswahllisten.

b) Die Sprachgenerierungsmodule von WING-NL erfüllen einen dezidierten Zweck im Rahmen eines Gesamtmodells zur Gestaltung softwareergonomischer Benutzungsoberflächen, das zu berücksichtigen ist.

Marx 1995 legt zwar letztendlich einen als „heuristisch“ bezeichneten Parser vor, der jedoch aus einer Diskussion linguistischer Theorien abgeleitet wird und weitgehendend theoretische Lösungsansätze der Computerlinguistik als Teilelemente nutzt. Daraus ergibt sich - verbunden mit den Notwendigkeiten des spezifischen Verwendungskontextes - eine theoretisch und empirisch fundierte Parsingstrategie. Die zu lösenden Probleme betreffen vor allem die Wahl eines Grammatikmodells, das Formproblem und die Integration des Korrekturmodus in den Generierungsprozeß.

- Grammatikmodelle für die Generierung

  Nach eingehender Überprüfung der in der formalen Linguistik vertretenen Grammatiktypen, die constraintbasierten, die Baumadjunktions-, die systemischen und die reversiblen Grammatiken, entscheidet sich Marx 1995 für eine FUG-Notation (functional unification grammar). Die Unifikation

wird dabei als ein Prozeß von Traversierung und Annotation des Grammatikbaums realisiert.

- Formproblem
  Das Formproblem thematisiert die Frage, in welcher sprachlichen Oberflächenform eine generierte Inhaltsstruktur ausgeben werden soll. In der menschlichen Kommunikation hängt sie stark vom pragmatischen Kontext und dem Gesprächsverlauf ab (z. B. Was ist schon bekannt? Was ist wichtig? Wie genau und detailliert soll die Antwort ausfallen?). Die heutigen state-of-the-art-Systeme zur Sprachgenerierung entscheiden jedoch letztlich intuitiv. Die Integration der Sprachgenerierung in die WOB-Umgebung eines Werkstoffinformationssystems zwingt dagegen zu einer stärkeren empirischen und theoretischen Begründung. Besonders interessant ist hier die Verknüpfung mit den Folgerungen aus der Computertalk-These (cf. Krause 1992). Sie besagt, daß sich die Benutzer in einer Mensch - Maschine - Dialogsituation schon deshalb anders ausdrücken (andere Wortwahl, andere Syntax) als in der zwischenmenschlichen Kommunikation, weil sie wissen, daß sie mit einem Computer interagieren. Diese Beobachtung ließ sich auch bei den empirischen WING-Tests nachweisen. Marx 1995 zieht daraus den Schluß, Computertalkelemente in die Generierung zu integrieren. Die Zustandsanzeige wird somit so formuliert, wie der Benutzer die Frage gegenüber einem Computer formulieren würde, nicht in der Form, die er gegenüber einem menschlichen Gesprächspartner verwendet. Daß die Computertalk-These auf der Basis empirischer Ergebnisse von der Analyse- auf die Generierungsebene gehoben wird, ist ein wichtiger, auch theoretisch interessanter Effekt.

- Integration des Korrekturmechanismus
  Auf die - erst einmal naheliegende - Erarbeitung eines Korrekturparsings, der vorgegebene Sprachstrukturen partiell verändert, verzichtet Marx 1995 zugunsten einer neuen Gesamtanalyse. Der zuerst versuchte Ansatz eines reinen Korrekturmechanismus bewährte sich nicht. Ein Nebeneffekt dieser Entscheidung ist, daß der gleiche Algorithmus für die alternative natürlichsprachlichen Eingabe ohne Beteiligung des graphischen Modus benutzt werden kann.

Mit WING-NL liegt somit für Softwareentwicklungen im Rahmen des WOB-Modells ein ausgearbeitetes Teilsystem vor, das zwar allein wegen des notwendigen Lexikonaufbaus jeweils neu an spezielle Anwendungsgebiete angepaßt werden muß, die prinzipielle Lösungsstrategie jedoch prototypisch vorgibt. Damit wird ein Einsatz auch im kommerziellen Umfeld möglich.

4.4.3 Korrigierbare, vereinfachte formalsprachliche Notation

Als dritte Alternative ließe sich eine vereinfachte SQL-Notation, wie ihn die HTM-Datenbank in Petten als reine Zustandsanzeige kennt, nach dem gleichen Verfahren wie die Lösung mit der natürlichsprachlichen Generierung aufbauen. Die Notation müßte sich als quasi-natürlichsprachlicher Formalismus auch ohne SQL-Kenntnisse interpretieren lassen und wäre durch den Benutzer veränderbar.

Noch offen bleibt, ob es gelingt, eine vereinfachte SQL-Syntax zu entwerfen, die algorithmisch eindeutig interpretierbar und gleichzeitig von Benutzern ohne SQL-Kenntnisse fehlerfrei veränderbar ist.

Bei Literaturrecherchen entspräche diese Lösung einer Notation, die sich an die heute üblichen kommandoorientierten Anfragesprachen wie MESSENGER anlehnt.

4.4.4 Fazit Zustandsanzeige mit Korrekturmodus

Die in Abschnitt 4.4 diskutierten Gestaltungstechniken lösen weitgehend das Problem einer dynamischen Rücknahme von Selbsterklärungsfähigkeit zugunsten von mehr Platz auf dem Bildschirm und kürzerer Interaktionswege (schnellere Bedienung) - ein altes Problem gegenläufiger Wünsche in der Softwareergonomie.

Sie realisieren mit den Alternativen in den Abschnitten 4.4.2 und 4.4.3 gleichzeitig eine begründete und empirisch bewährte, nichttriviale Modalitätsmischung.

4.5 Werkzeugmetapher und doppelte Interpretierbarkeit

Fassen wir nochmals die bisher in das WOB-Modell eingebrachten Gestaltungsstrategien zusammen, die den in Abschnitt 3.2 genannten, empirisch ermittelten Schwächen begegnen:

- Durch die konsequente Verwendung der dynamischen Anpassung / kontextsensitiven Durchlässigkeit aufeinander bezogener Felder, innerhalb der Bildschirmgrenzen und über sie hinaus, werden redundante Eingaben durch intelligentes Systemverhalten vermieden. Die darüber hinaus mögli-

che effiziente Platzverwaltung erhöht die Vorlageleistung und verkürzt die Bedienwege.

Gleichzeitig regelt die kontextsensitive Durchlässigkeit von Informationen zwischen den Modalitäten (bisher natürlichsprachlich und graphisch) den Informationstransfer von einer Modalität in die andere als Voraussetzung für nichttriviale multimodale Systeme.

- Die mehrfache Nutzung der natürlichsprachlichen Modalität (alternativ einer quasi-natürlichsprachlichen formalen Notation) als alternativen Eingabemodus, als Korrekturmodus und Zustandsanzeige erlaubt - genauso wie das Prinzip des komprimierten Eingangsbildschirms - eine dynamische Reduzierung der Selbsterklärungsfähigkeit mit wachsender Systemkenntnis. Der Benutzer tauscht an beliebiger Stelle und in beliebigen Umfang Platz zur Ergebnisdarstellung gegen nicht mehr benötigte Vorlageleistung.
- Der Formularmodus auf der Basis von Dialogboxen mit OK-Ausgang (statt handlungsorientierte (Teil-)Aktionen in den Dialogboxen), verbunden mit Navigationsleitlinien auf der Basis kognitiver Standardsuchsituationen, ermöglicht vor allem Anfängern ein weitgehend systemgesteuertes Arbeiten ohne Lern- und Entscheidungszwang. Gleichzeitig unterstützt der Formularmodus eine objektorientierte Sichtweise, die eine Metaphernbildung nicht voraussetzt.

Trotz dieser Maßnahmen enthält das WOB-Modell bisher noch zu wenig Flexibilität. Vor allem die Navigationsmöglichkeiten bleiben für fortgeschrittenere Benutzer zu starr. Der Ausgleich zwischen dem Wunsch, möglichst wenig über die Abfolge der einzelnen Parametrisierungschritte nachdenken zu müssen, was zu einer systemseitigen Vorgabe der Bedienabfolge führt, und dem entgegengesetzten Wunsch, die Abfolge der Parametrisierungsschritte selbst bestimmen zu wollen, ist noch unbefriedigend gelöst.

Flexibilität schafft bisher vor allem der jederzeit mögliche Modalitätswechsel und der natürlichsprachliche Korrekturmodus bzw. die variable Nutzung des komprimierten Eingangsbildschirms.

Deshalb soll nach neuen Möglichkeiten gesucht werden, der durch den Formularmodus eingebrachten Starrheit der Dialogführung zu begegnen, ohne deren Vorzüge für Anfänger zu verlieren. Allgemeiner ausgedrückt geht es darum, zwei generell als Widersprüche empfundenen Forderungen aufzulösen: die nach höchstmöglicher Flexibilität in der Navigation und gleichzeitig stärkerer Systemsteuerung.

Hierfür wird noch einmal der Grundgedanke multifunktionaler Werkzeugobjekte aus Abschnitt 4.2 aufgegriffen.

4.5.1 Die Schwächen mehrfunktionaler Werkzeuge in N/JOY und OS/2

Man tut N/JOY und dem workplace environment Gedanken von OS/2 unrecht, wenn man bei der Aufzählung der in Abschnitt 4.2 genannten Schwachstellen stehen-bleibt. Die Grundidee multifunktionaler Funktionsobjekte - oder anders ausgedrückt: die der mehrfunktionalen Werkzeuge - scheint vor allem den Ansprüchen fortgeschrittener Benutzer an eine flexible Navigation durchaus gerecht zu werden. Die Frage ist, ob sich Modifikationen finden lassen, die die fraglos gegebenen Schwächen zumindest abmildern, und ob sich solch ein verändertes Konzept mit dem Formularmodus widerspruchsfrei verbindet.

Um hierzu einen Ansatzpunkt zu finden, sollen nochmals zwei Zusatztechniken aufgegriffen werden, um die N/JOY das Konzept dezidierter Funktionsobjekte erweitert.

a) Werkzeuge, die einfunktional bleiben, sind zu schwach, um Menüstrukturen zu ersetzen. Dies wurde schon beim N/JOY Bleistiftkonzept klar, das die Schriftart und -größe festlegt. Wird das Werkzeug Bleistift auf der untersten Spezifikationsebene definiert, braucht jeder Benutzer eine Fülle von Bleistiften, für die er auf dem Bildschirm bald keinen Platz mehr hat.

   Die Reduktion auf eindimensionale Werkzeuge entspricht auch nicht unserer Werkzeugerfahrung. Es gibt zwar einfunktionale Werkzeuge wie den Reißwolf, daneben sind wir jedoch von einer Fülle komplexer Werkzeuge umgeben, die die verschiedensten Funktionen ausführen. Es gibt keine Waschmaschine, die nur das Wollsiegelprogramm beherrscht, sondern Waschmaschinen haben Bedienungselemente (Tasten, Knöpfe und Skalen), mit denen der Benutzer vor Beginn festlegt, welche spezifischen Parameter beim Start wirken sollen.

   In der elektronischen Welt entspricht dem eine Dialogbox, in der ein allgemeines Werkzeug auf spezifische Aktionen festgelegt werden kann. Z. B. enthält die dem Bleistift bei N/JOY zugeordnete Dialogbox eine Liste der Schriftarten, aus denen sich der Benutzer die aktuell gewünschte aussucht. Wird das Werkzeug angewendet, ist es auf diese Schrift „eingestellt“.

b) OS/2-Entwickler und das N/JOY-Entwicklungsteam würden die in Abschnitt 4.2 gegebene „natürliche“ Interpretation von drag & drop wohl ab-

lehnen. Für sie ist drag & drop ein abstraktes Auslösekonzept, das ein Werkzeug mit einem Objekt, auf das es wirken soll, verknüpft. Dem entspricht auch die Existenz des zweiten Auslöseweges: drag & drop kann durch eine Abfolge von Einfachklicks ersetzt werden.

Die Kritik in Abschnitt 4.2 war allerdings nicht, daß ein abstraktes Auslösekonzept nicht prinzipiell möglich wäre, sondern sie war empirischer Natur: Benutzer machen bei seiner Anwendung Fehler.

Die Abstraktheit des mehrfunktionalen Werkzeugkonzepts von N/JOY und OS/2 wird dadurch weiter unterstrichen, daß Werkzeuge auch selbst „betroffene“ Objekte sein können, auf die andere Werkzeuge wirken. So kann bei N/JOY ein Drucker auf den Kopierer gezogen werden, wodurch ein neuer Drucker entsteht. Die Farbpalette ist nicht nur dazu da, die Schriftfarbe oder die der Graphik festzulegen, sondern kann darüber hinaus das Erscheinungsbild des Druckers oder Papierkorbs auf der Bildschirmoberfläche verändern.

Diese Generalisierung fasziniert erst einmal. Sie hat jedoch ihren Preis. Der Benutzer muß vor jeder Aktion festlegen, welche Ikone Objekt und welche Werkzeug sein soll.

- Soll einer Ikone der Status '(betroffenes) Objekt' zukommen, wird sie *markiert*, soll sie Werkzeug sein (Funktionsstatus), *aktiviert*.

  Das betroffene Objekt läßt sich mit der linken Maustaste markieren und das Werkzeug anschließend durch Anklicken mit der rechten Maustaste aktivieren. D. h., daß das aktivierte Werkzeug das markierte Objekt verändert.

  Bei der alternativ möglichen drag & drop - Bedienung wird ein Objekt mit der linken Maustaste ergriffen und zum Werkzeug gezogen und dort losgelassen (z. B. Dokument über Reißwolf).

- Zusätzlich zur Statusfestlegung (aktivieren und markieren) muß es einen Weg geben, die Werkzeuge einzustellen, d. h., die zugehörigen Dialogboxen zu öffnen.

  Bei N/JOY öffnet ein Doppelklick der linken Maustaste alle Container-Objekte, die beliebige andere Objekte enthalten können. Alle Objekte der untersten Hierarchiestufe (unabhängig vom Status) werden durch Doppelklick mit der rechten Maustaste geöffnet.

Wie in Abschnitt 4.2 diskutiert, ergaben empirische Tests, daß der Benutzer durch die Vierfachdifferenzierung der Maustastenbedienung überfordert ist (cf. Weingärtner 1992). Er macht Fehler und wird durch ihre Komplexität von der eigentlichen Aufgabe abgelenkt. Dies ist ein gutes Beispiel

dafür, daß eine (überzogene) Generalisierung und ein theoretisch einfaches Grundmodell nicht zwangsläufig zu einer Erleichterung für den Benutzer führt. Bei dem N/JOY-Verfahren dürfte das Gegenteil der Fall sein.

Faßt man die Ergebnisse der erneuten Diskussion der Zusatztechniken von N/JOY zusammen, stellt sich für das WOB-Modell die Aufgabe, eine Interpretation der Werkzeugmetapher zu finden, die mehrfunktionale Werkzeuge zuläßt, ohne die natürliche Auslösetechnik zu opfern. Der Benutzer darf nicht mit der expliziten Statusfestlegung, einem komplexen Auslösemechanismus und der aus beiden erwachsenden, verwirrenden Vielfalt von Handlungsmöglichkeiten allein gelassen werden.

4.5.2 Werkzeugmetapher im WOB-Modell

Machen wir uns noch einmal die generellen Vorzüge der Werkzeugmetapher unter dem Blickwinkel der Objektorientierung klar: Die Metapher eines Werkzeugsystems bedient sich gegenüber dem Objekt-Funktions-Schema einer anderen Konzeptualisierung, die die Diskrepanz zur Sichtweise der objektorientierten Programmierung weitgehend vermeidet, aber trotzdem eine Parallele in der realen Umwelt hat. Bei ihr entsprechen Handlungen Werkzeugen, die diese Handlungen ausführen, und durch Ikonen symbolisiert werden (im geöffneten Zustand durch zugehörige Fenster). Diese Werkzeuge lassen sich wie die meisten technischen Geräte in unserer Umwelt auf spezifische Gegebenheiten einstellen, was einer Parametrisierung in einem Fenster entspricht, die der Benutzer durch 'OK' abschließt (Ende der Parametrisierung; Werkzeug kann jetzt eingesetzt werden).

Die Werkzeugobjekte sind dann im Sinne des Zitats zur natürlichen Sprache aus Abschnitt 4.2:

> „Objekte ... als mehr oder minder dauerhafte Gegebenheiten, die durch ihre verschiedenen momentanen Zustände hindurch im Wechsel ihrer Attribute ihre Identität bewahren.“ (Kutschera 1971:308f.)

Das Interessante am Werkzeugansatz für das WOB-Modell ist, daß damit das Denken in Objekt-Funktions-Beziehungen vermieden wird. Gleichzeitig bleibt er zur Konzeptualisierung in der objektorientierten Programmierung weitgehend kompatibel. Der Konzeptualisierungsunterschied besteht vor allem darin, daß das Werkzeugobjekt der BOF auf ein anderes Objekt einwirkt, statt daß eine Meldung an ein Objekt geschickt wird, das die Methoden bereits inhärent enthält. Für den Benutzer steckt die Funktionalität somit im

Werkzeug, nicht inhärent im betroffenen Objekt. Diese Differenz läßt sich jedoch bei der Modellbildung problemlos 1:1 umsetzen.

Der Konzeptualisierungsunterschied scheint als Abweichung zur Denkweise bei der objektorientierten Programmierung deutlich geringer als die Einführung expliziter Funktionsaufrufe. Deshalb wurde diese Interpretation der Objektorientierung für die BOF als strikte Objektorientierung bezeichnet.

Komplexe Werkzeugoberflächen im Sinne von N/JOY sind zwar sehr flexibel, mächtig und für fortgeschrittene Benutzer konzeptuell übersichtlich, überfordern aber in der Regel vor allem ihre gelegentlichen Benutzer durch den inhärenten Zwang zur Eigenverwaltung und Festlegung der Wirkungsrichtung der Objekte (Auf welches Objekt wirkt das Werkzeug, das in einem anderen Zusammenhang wieder selbst betroffenes Objekt eines Werkzeugs sein kann?).

Deshalb ist im WOB-Modell die Rollenverteilung zwischen Funktionsobjekt (= wirkendes Werkzeug) und betroffenem Objekt nicht frei, sondern systemseitig festgelegt.

Gleichzeitig überlagert die WOB-Abfragekomponente die Werkzeugstruktur mit einem Konzept der systemgesteuerten Werkzeugauslösung, das im Standardfall dem Benutzer die Entscheidung über das Öffnen oder Aktivieren eines Werkzeugs abnimmt. Öffnet sich ein Werkzeug automatisch, entspricht dies einer Systemführung, die Bedienschritte einspart, wenn ein Teilschritt in einem bestimmten Aufgabenkontext zwingend vorgegeben ist.

Soll ein bestimmtes Werkzeug in einer bestimmten Anwendungssituation dem Benutzer zwar erkennbar zur Verfügung stehen, aber geschlossen bleiben, da der Einsatz des Werkzeugs nicht zwingend vorgegeben ist, wird das WOB-Konzept der Voraktivierung wesentlich. Folgende *Zustände* der Werkzeugobjekte sind unterschieden:

a) Nicht aktiviert

Geöffnete Fenster treten in den Hintergrund (ohne Schließen); die Titelleiste und der Fensterhintergrund werden grau. Die Ikonen bleiben ohne Hervorhebung.

Nicht aktive, offene Fenster werden durch das Anklicken der Überschriftszeile oder eine beliebige Aktion im nicht aktiven Fenster (z. B. Cursor in Datenfeld setzen oder Listenelement anklicken) aktiviert und gleichzeitig in den Vordergrund gestellt.

Der Benutzer kann Werkzeugikonen an jeder beliebigen Stelle des Dialogablaufs durch Doppelklick mit der linken Maustaste öffnen und spezifisch parametrisieren. Die Einstellungen werden erst wirksam, wenn das Werkzeug aktiviert wird.

b) Aktiviert

Geöffnete Fenster lassen sich nur in diesem Zustand bearbeiten.

Sie sind durch eine farbliche Hinterlegung der Titelleiste und des Fensterhintergrundes deutlich akzentuiert von nicht aktiven, geöffneten Fenstern zu unterscheiden.

Die geöffneten Fenster aktiver Ikonen werden entsprechend der Farbsymbolik ihres Fensters farblich hinterlegt.

Die Aktivierung erfolgt bei nicht aktiven Ikonen durch Doppelklick mit der linken Maustaste (bei voraktivierten auch durch Einfachklick; cf. Punkt c) unten). Das Fenster öffnet sich.

Aktivierung bedeutet, daß das Werkzeug vom Benutzer parametrisiert werden kann. Ausgelöst wird das Werkzeug erst mit dem Bestätigen der Parametrisierung durch das Auslösen der 'OK'-Taste, wenn das Werkzeug an dieser Stelle des Dialogs ablauffähig ist. Letzteres ist an der Voraktivierung erkenntlich.

Als Variante läßt sich in allen Werkzeug-Dialogboxen, die aktiviert werden, ohne im Dialogablauf voraktiviert zu sein, zusätzlich eine Taste 'Automatisch aktivieren' einführen. Sie würde dafür sorgen, daß das Werkzeug im nächsten Zustand der Voraktivierung automatisch ausgelöst wird. Der Benutzer würde sich dann den Doppelklick auf die voraktivierte Ikone ersparen (cf. den folgenden Punkt c).

Diese Technik ist insofern riskant, als der Benutzer genau wissen muß, wann das Werkzeug voraktiviert wird, um es nicht an der falschen Stelle im Dialogablauf wirken zu lassen.

c) Voraktivierung

Ikonen, die vom Benutzer in bestimmten Dialogsituationen aktivierbar sind (aber sich nicht selbsttätig öffnen, d. h. keine automatische Aktivierung), werden durch eine relativ unauffällige Rasterhinterlegung in der Farbsymbolik des zugehörigen Fensters gekennzeichnet.

Die benutzergesteuerte Aktivierung erfolgt durch Einfachklick auf die voraktivierte Ikone. Die Dialogbox zur Parametereinstellung öffnet sich. Verläßt der Benutzer über 'OK' die Dialogbox, wird die Aktion ausgeführt.

Will der Benutzer die bestehende Parametrisierung des voraktivierten Werkzeugs übernehmen und die Dialogbox umgehen, führt ein Doppelklick zur direkten Aktivierung des Werkzeugs.

Der fortgeschrittene Benutzer verwaltet somit im WOB-Grundmodell - im Gegensatz zum Verfahren in N/JOY - nur zwei statt vier Mausaktionen. Der Anfänger und gelegentliche Benutzer kommt mit dem Einfachklick der linken Maustaste aus.

- Doppelklick links zur Parametrisierung der Werkzeuge, solange das Werkzeug nicht aktiviert ist.

  Dem liegt eine Änderung der durch den Systemstandard vorgegebenen Abarbeitungsreihenfolge zugrunde, auf die der Anfänger und gelegentliche Benutzer verzichten kann.

- Einfachklick links auf voraktivierte Werkzeuge zur Auslösung der aktuell gültigen Werkzeugaktion über das Öffnen der Dialogbox, die mit 'OK' verlassen wird.

  Im Sinne eines fehlertoleranten Systems sollte hier keine andere Reaktion erfolgen, wenn der Benutzer den Zweifachklick benutzt. Der Preis für diese Fehlertoleranz ist, daß eine Verkürzungsmöglichkeit entfällt: Beide Einfachklicks mit der linken Maustaste ließen sich sonst zu einem Doppelklick zusammenziehen. Die Dialogbox bliebe beim Doppelklick geschlossen, die Aktion würde sofort mit den voreingestellten Parametern durchgeführt.

  Bei der fehlertoleranten Lösung muß der Benutzer zweimal den Einfachklick ausführen bzw. (schnell hintereinander möglich) zweimal ENTER drücken.

Die vorgegebene Standardeinstellung, die durch einen Systemadministrator verän-derbar ist, legt somit das Zusammenspiel der sich öffnenden Fenster und die in bestimmten Dialogsituationen zugelassene Werkzeugauswahl (über die Voraktivierung) fest. Da die Möglichkeiten der flexiblen Anpassung des Systems an Sonderbedürfnisse für den erfahrenen Benutzer durch die Grundphilosophie des Werkzeugsystems erhalten bleiben (z. B. Werkzeuge an beliebiger Stelle im Dialog parametrisieren, nicht nur vor der Auslösung), verbinden sich maximale Flexibilität für den fortgeschrittenen Benutzer mit einer einfachen, stärker systemdeterminierten Abarbeitung der Suchanfrage, ohne daß ein Widerspruch entsteht.

Nicht manipulierbar ist dagegen die allgemeine Bezugsstruktur: Alle Objekte, die keine Dokumente, Dokumentgruppen oder Suchobjekte sind, wirken als mehrfunktionale Werkzeuge auf die Such- oder Ergebnisobjekte. Die Festlegung, welches Objekt Funktionsobjekt und welches betroffenes Objekt ist, bleibt stets systemseitig determiniert. Die Determinierung kann auch der fortgeschrittene Benutzer nicht aufheben. Er kann nur die Parameter voreinstellen, den Status des Werkzeugs beeinflussen (automatisch aktivieren etc.) und seinen Zustand ändern (aktiv/nicht aktiv).

Diese Einschränkungen, zusammen mit den im folgenden angeführten Zusatzmerkmalen wie dem der doppelten Interpretierbarkeit, versetzt den Benutzer wieder in die Lage, mit seiner Gestaltungsfreiheit sinnvoll umgehen zu können. Er behält die Übersicht und wird durch von ihm geforderte Detailentscheidungen von seiner inhaltlichen Fragestellung nicht mehr übermäßig abgelenkt.

Fazit: Somit bieten sich zwei objektorientierte Interpretationen für das WOB-Modell an, die des Formulars und die des multifunktionalen WOB-Werkzeugs. Beide sind bereits weitgehend von handlungsorientierten Elementen befreit, was eine generelle objektorientierte Sichtweise unterstützt.

Das multifunktionale WOB-Werkzeug ist im Gegensatz zum Formularmodus eine Systemmetapher. Anders als z. B. bei der Bürometapher fällt jedoch ihre Abstraktheit auf. Semantisch übertragen werden keine Details, sondern der funktionale Kern des Wirkens von Werkzeugen. Diese Art der Semantik merkt sich der Benutzer nicht als Liste von Details, er begreift sie als Wirkprinzip.

Deshalb gehen auch die Angriffe von Nardi/Zarmer 1993 gegen das Metaphernkonzept graphischer BOF für die WOB-Werkzeugmetapher ins Leere (zur Metaphernkritik allgemein cf. Krause 1996):

> „... metaphors with their simplicity and imprecision cannot hope to carry the semantic burden. ... [Halasz/Moran 1981 and Johnson 1987] have pointed out that metaphors are bound to be incomplete representations of the systems they are meant to expose. ... iconic database language ... cannot afford such laxity. It must be based on an underlying system of precise semantics, a system that is internally consistent and complete. ... [Halasz/Moran 1981] noted that people usually can easily derive one basic meaning from a metaphor, but have difficulty, ‘sort[ing] out the relevant mappings and allowable inferences’. It is precisely these mappings and relevant inferences that we want to cap-

> ture in exact ... Metaphors are slippery things, and not just because they contain relevancies (do we set the trash can out for garbage pick-up on Friday morning?) and incompleteness ... More than that, we give them more responsibility than they can handle - they are like very clever children of whom we end up asking much to much, because they seem so bright and able.“ (Nardi/Zarmer 1993:18f.)

Beide WOB-Prinzipien, die des Formulars und die des multifunktionalen WOB-Werkzeugs, scheinen zusammen die widersprüchlichen Anforderungen an Flexibilität und voreingestellte Dialogleitlinie zu erfüllen, was jedoch wenig nutzt, solange zwischen ihnen keine Verbindung besteht. Bereits in Abschnitt 3.2.2 wurde der Vorschlag als zu inflexibel abgelehnt, getrennte Oberflächen für Anfänger und Fortgeschrittenen gleichzeitig und parallel in einem System zur Verfügung zu stellen.

Der folgende Abschnitt führt die konzeptuelle Verbindung zwischen beiden Sichtweisen ein. Sie ist der eigentliche Kern des WOB-Modells.

4.5.3 Doppelte Interpretierbarkeit

Bei der Analyse der generellen Vor- und Nachteile des Formularmodus für die Datenbankabfrage zeigten sich gegenläufige Wünsche und Bedürfnisse, zu denen die Forderungen nach höchstmöglicher Flexibilität in der Navigation und der gleichzeitige Wunsch einer stärkeren Systemsteuerung gehören.

Die Erweiterung des WOB-Modells um das Konzept der doppelten Interpretierbarkeit versucht, diesen Nachteilen mit einer Strategie zu begegnen, die beim Aufbau des mentalen Benutzermodells ansetzt. Jeder Benutzer baut sich ein mentales Modell der verwendeten Software auf. Je konsistenter es ist und je weniger Widersprüche die daraus ableitbaren Handlungen zu den Systemreaktionen zeigen, um so fehlerfreier arbeitet er mit dem Computersystem. BOF müssen deshalb so gestaltet werden, daß das „richtige“ mentale Modell induziert wird.

Je komplexer andererseits die Anforderungen an ein System werden, um so schwieriger wird es in der Regel, ein korrektes mentales System aufzubauen.

Menschen entwickeln zur Handhabung der wachsenden Komplexität in ihrer Umwelt allerdings interessante Gegenstrategien, die man sich an der sog. „naiven Physik“ klarmachen kann (cf. Owen 1986, Hayes 1978).

Jeder Mensch braucht bestimmte Vorstellungen über physikalische Gesetzmäßigkeiten, um sich, ohne körperlich Schaden zu nehmen, in seiner Umwelt bewegen zu können. Vergleicht man diese mentalen Modelle mit denen von Physikern, zeigt sich nicht nur eine Reduzierung der physikalischen Regeln, sondern es lassen sich Abweichungen feststellen, die mit dem Physikermodell nicht mehr kompatibel sind. Die „naiven“ Regeln reichen jedoch aus, um sich physikalische Phänomene der Umwelt im täglichen Leben erklären zu können und, ohne Schaden zu nehmen, darauf zu reagieren. Läßt sich jemand später zum Physiker ausbilden, muß er das „naive Physikmodell“ durch ein komplexeres Modell ersetzen, das dann allerdings auch Erklärungen für physikalische Phänomene liefert, die dem Laien nicht mehr zugänglich sind.

Das WOB-Modell versucht - ansetzend beim Phänomen der „naiven Physik“ - einen Designentwurf zu finden, der sich *doppelt* interpretieren läßt. Im Gegensatz zum obigen Beispiel ist allerdings die einfachere Interpretation in der komplexeren enthalten. Es sind zwei Sichtweisen, die untereinander voll kompatibel sind.

Der Benutzer ohne EDV-Hintergrundwissen, der sich diese Kenntnisse eventuell auch deshalb nicht aneignen möchte, weil er das System nur gelegentlich benutzt, soll ein einfaches mentales Modell aufbauen können, das in bezug auf die Bedienung der Oberfläche ausschließlich Vorstellungen erfordert, die für die Entwicklung eines mentalen Modells für Standardsysteme mit graphisch-direktmanipulativer BOF notwendig sind. Anders ausgedrückt heißt dies: Zum Aufbau des mentalen Modells ist kein größerer kognitiver Aufwand nötig als bei herkömmlichen Maskensystemen.

Der Anfänger und gelegentliche Benutzer braucht im wesentlichen nur Formulare zu interpretieren und die Felder zu kennen, in denen er Werte parametrisiert. Er agiert im Formularmodus nach dem in Abschnitt 4.3 beschriebenen Muster. Daß sich hinter den Bildern des Eingangsbildschirms und den geöffneten Formularfenstern mehr verbirgt, daß damit der gesamte Suchprozeß flexibel gestaltet werden kann, dieses Wissen ist nicht notwendig, um ein kohärentes mentales Modell für die systemgesteuerten Standardsuchsituationen aufzubauen.

Oder aus der Sicht der Werkzeugmetapher her ausgedrückt: Die geöffneten Fenster der bisher eingeführten Formularobjekte des WOB-Modells lassen sich ohne Rekurs auf das Modell eines objektorientierten Werkzeugansatzes als Formulare verstehen, die einen Suchauftrag an die Datenbank spezifizieren.

Für die von fortgeschrittenen Benutzern geforderte Flexibilität reicht dieses mentale Modell nicht aus. Der fortgeschrittene Benutzer soll bewußt erkennen, daß die geöffneten Suchobjekte geöffnete Werkzeugikonen sind, die sich vor dem Hintergrund der WOB-Werkzeugmetapher im Sinne eines strikt objektorientierten Systemansatzes handhaben lassen.

4.6 Iteratives Retrieval und graphisches Ergebnisretrieval

In Abschnitt 3.2 wurde auf das Phänomen hingewiesen, daß Benutzer in einigen Bereichen mit visuellen Darstellungen wie Planskizzen oder Liniengraphiken „weiter-denken". In WING-M2 wird z. B. die Liniengraphik selbst als Modus zur Formulierung der weiteren Suchabsicht benutzt. Der Zwang, die Suchabsicht wieder als alphanumerisch auszudrückende Parametermenge in der strukturierten Umgebung des graphisch-direktmanipulativen Ersteinstiegs zu formulieren, ist in dieser Situation genauso „unnatürlich" wie der natürlichsprachliche Abfragemodus.

In diesem Fall bestimmt der Modus der Ergebnisdarstellung die für die Weiterführung der Suche optimale Modalität.

Im WOB-Modell ist diese Komponente als spezielles Werkzeug - als Ergebnistransformator - realisiert, das zudem die einfache Umwandlung einer Ergebnisdarstellung in eine andere (auch ohne weitere Suchabsicht) leistet.

4.6.1 Der Ergebnistransformator ETRANS

In der Sichtweise des WOB-Werkzeugsystems sind Ergebnisbildschirme geöffnete Fenster von Objekten, die durch die Evaluierung eines Suchobjekts (d. h. die Anwendung des Suchobjekts als Funktionsobjekt auf das Objekt Datenbank) neu entstehen. Suchobjekt und Ergebnisobjekte bilden ein Objektpaar, das durch die Evaluationsrelation (EVAL) miteinander verbunden ist.

Eine weitere Besonderheit ergibt sich daraus, daß Ergebnisfenster ihren Typ wechseln können. Sie lassen sich mit Hilfe eines speziellen Werkzeugs zu einem Suchobjekt transformieren, das strukturgleich zur Ergebnisliste ist und dessen Anfangsparametrisierung dem Ergebnisbildschirm entstammt. In der Sichtweise der Objektorientierung entsteht somit wiederum ein neues Objekt, das in einer speziellen, formal definierbaren Relation (TRANS) zum Ergeb-

nisfenster steht. Da sich jedes Ergebnisfenster wiederum durch die Evaluationsrelation mit dem vorangegangenem Suchobjekt verbinden läßt, ergibt sich eine Interpretation des iterativen Retrievals unter (auch formaler) Einbeziehung der vorangegangenen Ergebnisse, das sich als Kette von Objektumwandlungen definiert, bei der sich die Relationen EVAL und TRANS abwechseln.

Die Etablierung der EVAL-Relation wird durch die zugrundeliegende Datenbankmaschine geleistet (bei WING-M2 durch COMFOBASE bzw. ORACLE); TRANS ergibt sich durch die Anwendung des Transformator-Werkzeugs ETRANS (als Funktionsobjekt) auf das Ergebnisfenster.

Je nachdem welcher Typ von Ergebnisfenster transformiert wird, unterscheiden sich die ausgeführten Aktionen. Diese parameterabhängige Definition von Aktionen ist ein wesentlicher Bestandteil der Mächtigkeit des Ansatzes der objektorientierten Programmierung und läßt sich in die objektorientierte Gestaltung von BOF übertragen.

Generell bedingt die Sichtweise von Ergebnisfenstern als neu geschaffene Objekte, daß sie am Hauptbildschirm als Ikonen abgelegt werden können und damit auch außerhalb der zeitlichen Reihenfolge der iterativen Suche zur Verfügung stehen.

Damit kann der Benutzer ein Ergebnis, das ein Zwischenresultat seiner Problemlösung ist, erst einmal abspeichern, sich einem neuen Problemkreis zuwenden und später auf das Ergebnis des ersten Suchproblems zurückkommen, ohne den schon erreichten Zustand nochmals eingeben zu müssen.

Die Abspeicherung von Ergebnisfenstern erfolgt prinzipiell zusammen mit der Abfrageformulierung.

Bei der Transformation eines Ergebnisfensters in ein Suchobjekt muß vermieden werden, daß sich der Benutzer über den Typ der Objekte im Unklaren ist. Deshalb wird mit einer klaren Farbsymbolik gearbeitet.

4.6.2 Ergebnismodus Query by example (QBE)

Der QBE-Ergebnismodus ist die einfachste Art der Ergebnistransformation im WOB-Modell.

QBE ist ein recht früh entwickelter (Zloof 1983) und kommerziell genutzter Grundtypus einer „natürlichen“ BOF für Fakteninformationssysteme und

mittlerweile als Zugang zu tabellarisch organisierten Datenbanken weit verbreitet (cf. z. B. PARADOX oder eingeschränkt auf eine Tabelle bei Q&A).

Die Grundidee von QBE ist einfach. Dem Benutzer werden die ausgewählten Tabellen der Datenbank graphisch am Bildschirm vorgelegt. Er trägt in die betreffenden Felder Werte und Restriktionen ein, die die Ergebnisauswahl steuern. Damit wird ein Großteil der SQL-Syntax, der die Zuordnung von Werten und Bedingungen zu den Tabellen regelt, überflüssig.

Setzt man QBE in Beziehung zu den bisher diskutierten Grundtypen der Retrievalgestaltung, sind zwei Faktoren wichtig:

- QBE basiert auf dem Ausfüllen von Tabellen. Tabellen sind eines der prototypischen Beispiele für visual formalisms, die Nardi/Zarmer 1993 anführt (cf. Abschnitt 4.3.1 und Krause 1996). Sie lassen sich als Untergruppe von Formularen im Sinne des WOB-Modells ansehen.
- Sobald bei der Formulierung der QBE-Suchanfrage die Join-Operation ins Spiel kommt, gilt das gleiche wie für kommandoorientierte SQL-Anfragen: Der Benutzer muß EDV-motiviertes Wissen einsetzen. Daß dieser Bruch deutlich als störend empfunden wird und zu Fehlern führt, zeigten die Pretests mit einem QBE Prototypen im Projekt WING-IIR (cf. Lickleder 1990).

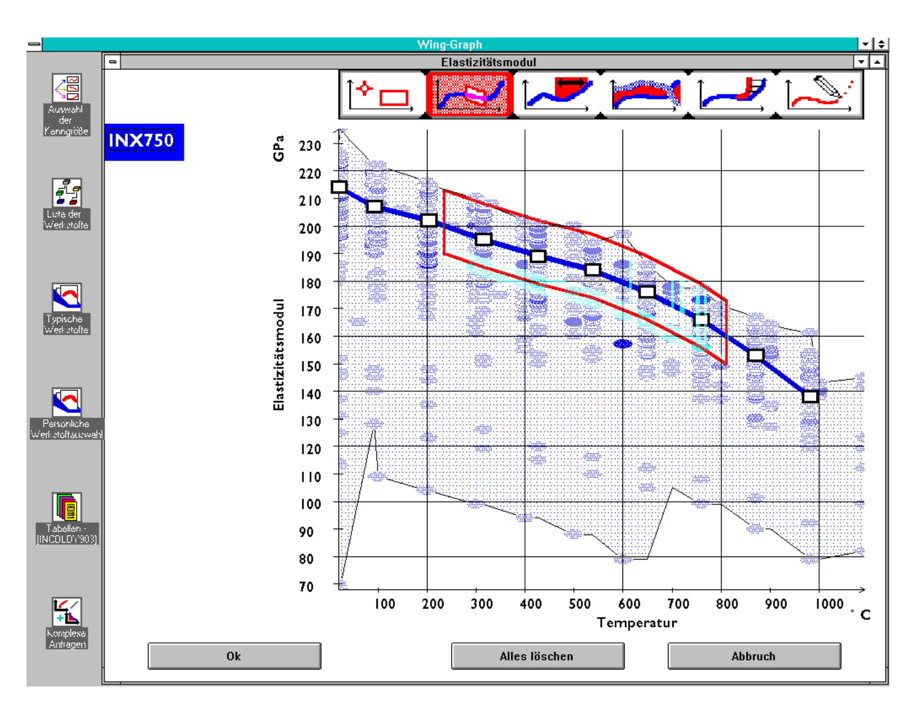
Im WOB-Modell wurde den empirischen Ergebnissen dadurch Rechnung getragen, daß der QBE-Modus auf die Ergebnistabellen eingeschränkt blieb. Der Ersteinstieg erfolgt über die bisher geschilderten Verfahren. Hat der Benutzer als Ergebnis eine Tabelle erhalten, kann er über den Ergebnistransformator die erhaltene Tabellenstruktur zur Neuformulierung (bzw. Modifikation) seiner Suchabsicht nutzen, wobei ihm alle konzeptuellen Mittel zur Verfügung stehen, die QBE für einfache Tabellen (ohne Join) kennt.

4.6.3 Graphisches Ergebnisretrieval

Engte Abschnitt 4.6.3 eine bekannte Retrievalstrategie ein und wies ihr im WOB-Modell ihren spezifischen Platz zu, führt die zweite Ergebnistransformation einen prinzipiell neuen Modus der Suche ein. Er wurde bisher anhand der Anforderungen in WING-M2 für die Werkstoffsuche beispielhaft entwikkelt (Modul WING-GRAPH) und ist in Wolff 1996 theoretisch auf der Basis einer Weiterentwicklung des Pinker-Modells (Pinker 1990) begründet und beschrieben.

Die im Kontext von WING-M2 erarbeitete visuelle Manipulationssprache für die Liniengraphik verdeutlicht die Grundideen dieses Designelements des WOB-Modells. Sie gilt in ihrer Typisierung jedoch nicht für alle WOB-Anwendungsmöglichkeiten, wenn sie auch die Benutzerkonzeptualisierung der Werkstoffsuche adäquat unterstützt. Erste Versuche mit einer Übertragung der visuellen Manipulationssprache auf den Bereich von Wirtschaftsdaten zeigten, daß die Grundelememte für einen generalisierenden Ansatz nicht ausreichen (cf. Wolff 1996:Abschnitt 8.2).

Vor den Hintergrund dieser Einschränkung soll im folgenden anhand des exemplarischen Beispiels der Werkstoffsuche die Grundidee des graphischen Ergebnisretrievals verdeutlicht werden.



Graphisches Ergebnisretrieval aus Wolff 1996:3

Im obigen Beispiel hat ein Benutzer als Ersteinstieg den Werkstoff INX750 ermittelt, der bestimmte Anforderungen an die Elastizität bei bestimmten Temperaturen erfüllt. Der Benutzer betrachtet zur Auswertung seines Ergebnisses das Elastizitätsmodul als Kurvenverlauf und kann z. B. zu folgenden Einschätzungen kommen (cf. Wolff 1996):

a) Die Kurve eignet sich nicht, weil sie im Hochtemperaturbereich zu weit unten verläuft.

b) Der Werkstoff ist prinzipiell geeignet, aber zu teuer. Ich bräuchte somit einen anderen Werkstoff, der zwischen 200 und 800 Grad mit einer Abweichung von 10 Gpa nach oben oder unten der Kurve des Elastiztätsmoduls von INX750 entspricht.

c) Die Kurve müßte im Bereich ab 600 Grad über der von INX750 liegen.

Selbstverständlich könnte der Benutzer versuchen, obige kognitiven Überlegungen in die Parametereinstellungen der Modalität des graphisch-direktmanipulativen Ersteinstiegs umzusetzen und so über mehrere Iterationsstufen zum erwünschten Ergebnis zu kommen.

Dieser Modalitätswechsel von der kognitiven Vorstellung des Benutzers in einer bestimmten Phase des Retrievalprozesses zur Modalität, die technisch durch das benutzte EDV-Werkzeug erzwungen wird, unterscheidet sich jedoch in nichts von dem Zwang, sich an Computernormen anzupassen, die z. B. formale Anfragesprachen wie SQL dem Benutzer auferlegen.

Deshalb muß eine visuelle Manipulationssprache gefunden werden, die es erlaubt, die als Ergebnisdarstellung erzeugte Liniengraphiken ohne Modalitätsbruch umzuformen und als neue Suchformulierung zu benutzen.

Beim WOB-Modell wird in WING-GRAPH die Submetapher einer Parametrisierung des Mauscursors für die Realisierung der visuellen Manipulationssprache benutzt. Die oberhalb der Liniengraphik angezeigten Symbole stellen den Mauscursor ein - wie die Dialogboxen die WOB-Werkzeuge. Bezogen auf das obige Beispiel heißt dies:

a) Mit dem fünften Symbol läßt sich der Mauscursor so modifizieren, daß er die Kurve direktmanipulativ im hohen Temperaturbereich nach oben ziehen kann.

b) Das zweite Symbol stellt dem Mauscursor so ein, daß sich um die Kurve ein Band legt (hier: 10 Gpa nach oben und unten von 200 - 600 Grad), durch das die Kurven der Werkstoffe laufen sollen.

c) Mit dem dritten Symbol läßt sich der Mauscursor in seiner Wirkungsweise so modifizieren, daß er einen Bereich über der Kurve festlegt, durch den die Elastizitätsmodul-Kurven der gewünschten Werkstoffe laufen sollen.

WING-GRAPH sieht bisher sechs Suchtypen vor, die der Benutzer auch kombinieren kann. Sie sind spezifisch für die Werkstoffsuche und dort empirisch getestet. Andere Bereiche dürften weitere Suchtypen nahelegen, so daß nach einigen Generalisierungsversuchen mit einer standardisierten Auswahl

von Suchtypen zu rechnen ist, die den maximalen Umfang der Komponente graphisches Ergebnisretrieval des WOB-Modells festlegen würden. Bei jeder Anwendung wäre die adäquate Anzahl von Suchtypen hieraus empirisch abzuleiten.

4.6.4 Einordnung

In einem generelleren Zusammenhang ist das Phänomen des Modalitätsbruchs beim graphischen Ergebnisretrieval und seine Lösung im Rahmen des WOB-Modells aus zwei Gründen besonders interessant:

- Es zeigt die Grenzen des heute üblichen Paradigmas graphisch-direktmanipulativer BOF. Die zur Verfügung stehenden Darstellungsmittel sind nicht in allen Fällen ausreichend. Graphisch-direktmanipulative BOF dürften in naher Zukunft zumindest in einigen Bereichen durch weitergehende Visualisierungen ersetzt werden.

- Das graphische Ergebnisretrieval führt zu einem zweiten Beispiel (neben der Mischung mit dem natürlichsprachlichen Modus bei der Zustandsanzeige mit Korrekturmodus in Abschnitt 4.2.4) für eine plausible und durch kognitive Theorien zu stützende Modalitätsmischung. Zwei alternative Modalitäten lassen sich hier im iterativen Suchzyklus begründet und nichttrivial miteinander verbinden.

5 Fazit WOB-Modell

Das WOB-Modell scheint derzeit so weit gediehen, daß es sich - im Sinne eines Mittel-Modells von Abschnitt 2 - als Grundlage einer softwareergonomischen Oberflächengestaltung im Kontext graphisch-direktmanipulativer Systeme einsetzen läßt.

Es liefert u. a. Lösungen für einige altbekannte Widersprüche zwischen softwareergonomischen Forderungen wie der nach hoher Selbsterklärungsfähigkeit für Anfänger und einer flexiblen, schnellen Bedienung für Fortgeschrittene durch die dynamische Rücknahme der Selbsterklärungsfähigkeit und seine doppelte Interpretierbarkeit. Es zeigt weiter Ansatzpunkte für eine adäquate Interpretation der strikten Objektorientierung bei BOF und für nichttriviale, auch kognitionstheoretische begründbare Modalitätsmischungen zwischen der graphisch-direktmanipulativen Standardform, visuellen Dar-

stellungen wie der von Liniengraphiken und dem natürlichsprachlichen Modus.

Damit bietet es gleichzeitig den Einstiegspunkt in Überlegungen zu zukünftigen Oberflächen, in denen eine erweiterte Visualisierung die heutigen Standardtechniken graphisch - direktmanipulativer Bedienung mit ihren Ikonen, Listen, Mehrfachauswahlboxen u. ä. zurückdrängen dürfte. Wie das WOB - Modell diese Tendenzen verstärkt integrieren kann, ist das Thema von Krause 1996.

6 LITERATUR

Bates, M.J. (1989): The Design of Browsing and Berrypicking Techniques for the Online Search Interface. In: Online Review, Vol.13, No.5, S. 407-424.

Beimel, J., Hüttner, J., Wandke, H. (1992): Kenntnisse von Programmierern auf dem Gebiet der Software-Ergonomie. Vortrag Fachtagung "Arbeits-, Betriebs- und Organisationspsychologie vor Ort". 25.-27.5.1992. Bad Lauterbach.

Booch, G. (1994): Object Oriented Analysis and Design. Cummings, Redwood, CA.

Cohen J. (1991): The Unofficial Guide to the Workplace Shell. Verbreitet über Internet.

Cox, B.J.; Novobilski, A.J. (eds.) (1991): Object-Oriented Programming. An Evolutionary Approach. Addison-Wesley, Reading, MA, S. 1-68.

Fox, T. (1993): Kognitiv ergonomische Benutzerschnittstellen - Entwicklung interaktiver 3D-Visualisierungen und multimedialer Simulationen: Das TUTOR-System COCARD zur Einführung in Ultraschall-Untersuchungen des Herzens. GMD-Studien Nr. 218. GMD Sankt Augustin, S. 77.

Durding, B.M.; Becker, C.A.; Gould. J.D. (1977): Data Organization. In: Human Factors 19, S. 1-14.

Eberleh, E., Oberquelle, H. Oppermann, R. (Hrsg.) (1994): Einführung in die Software-Ergonomie. Gestaltung graphischer-interaktiver Systeme: Prinzipien, Werkzeuge, Lösungen. (Mensch Computer Kommunikation - Grundwissen; 1) 2. Aufl. Berlin, New York.

GSM (1995): GSM Online Styleguide Version 1.3 für Windows. Stuttgart.

Halasz, F; Moran, T. (1981): Analogy considered harmful. In: Proceedings Human Factors in Computing Systems Conference. Gaithersburg, Maryland, S. 383-386.

Hartmann, P. (1952): Einige Grundzüge des japanischen Sprachbaues. Heidelberg.

Hayes, P.J. (1978): The Naive Physics Manifesto. In: Michie, D. (ed.): Expert Systems in the Microelectronic Age. Edinbourgh.

Helander, M. (ed.) (1988): Handbook of Human-Computer Interaction. Amsterdam et al.

ISO 9241 (1991): Ergonomic Requirements for Office Work with Visual Display Terminals, Part 10, Dialogue Principles, Committee Draft, September 1991.

Ingwersen, P. (1994): Information Science as a Cognitive Science. In: Best, H. et al. (Hrsg.): Informations- und Wissensverarbeitung in den Sozialwissenschaften. Opladen, S. 23-56.

Johnson, J. (1987): How faithfully should the electronic office simulate the real one? In: SIGCHI Bulletin 19, S. 21-25.

Koch, H.; Noé, W. (1980): Mensch-Maschine-Beziehungen in Kommunikationssystemen. Ergonomische Grundlagen und Gestaltungsregeln für Konstruktion, Entwicklung und Projektierung. Siemens München. S. 55.

Krause, J. (1992): Natürlichsprachliche Mensch-Computer-Interaktion als technisierte Kommunikation: Die Computer-Talk-Hypothese. In: Krause, J., Hitzenberger, L. (Hg.): Computer Talk. Hildesheim: Olms. (Sprache und Computer; Bd. 12). S. 1-29.

Krause, J. (1993a): A Multilayered Empirical Approach to Multimodality:Towards Mixed Solutions of Natural Language and Graphical Interfaces. In: Maybury, M. (ed.): Intelligent Multimedia Interfaces. AAAI Press/MIT, Menlo Park, CA. S. 328-352.

Krause, J. (1993b): Komplexe Menüs, Häkchen und Funktionsobjekte. In: Rödiger, K.H. (ed.): Software-Ergonomie '93. Von der Benutzungsoberfläche zur Arbeitsgestaltung. Teubner, Stuttgart.

Krause, J. et al. (1993/94): Multimodality and Object Orientation in an Intelligent Materials Information System. In: Journal of Document and Text Management, Part 1: Vol.1, No.3, S. 256-275; Part 2: Vol.2, No.1. (to appear).

Krause, J. (1994): Das WOB-Modell. Informationswissenschaft Regensburg. Projekt WING-IIR. Arbeitsbericht 53. Juli 1994, Universität Regensburg.

Krause, J. (1996): Visualisierung und graphische Benutzungsoberflächen. Arbeitsbericht Informationszentrum Sozialwissenschaften (IZ), Bonn und Universität Koblenz-Landau, Institut für Informatik, Koblenz.

Krause, J., Womser-Hacker, C. (Hrsg.) (1990): Das Deutsche Patentinformationssystem. Entwicklungstendenzen, Retrievaltests und Bewertungen. Köln et al.

Krause, J., Womser-Hacker, C. (Hrsg.) (1996): Vages Information Retrieval und graphische Benutzungsoberflächen - Beispiel Werkstoffinformation. Schriften zur Informationswissenschaft, Universitätsverlag. Konstanz.

Krause, J.; Mandl, T.; Stempfhuber, M. (1996): Design des ersten Prototypen des ZVEI-Verbandsinformationssystems ELVIRA. Informationszentrum Sozialwissenschaften Bonn, Projekt ZVEI-Verbandsinformationssystem, Arbeitsbericht 3.

Kutschera, F. von (1971): Sprachphilosophie. München.

Lickleder, G. (1990): Der Zugriff auf Werkstoffdatenbanken mit der Abfragesprache Query-By-Example (QBE) am Beispiel der Werkstoffdatenbank von mtu. WING-IIR Arbeitsbericht Nr. 10, November 1990. LIR Regensburg.

Marx, J. (1991): Benutzertests zur natürlichsprachlichen Komponente von WING-IIR. Testaufbau, Ergebnisse und Anforderungsprofil für den Parser. WING-IIR Arbeitsbericht Nr. 18, Juli 1991. LIR Regensburg.

Marx, J. (1993): Die Benutzertests zu WING-M2. WING-IIR Arbeitsbericht Nr. 43, Dezember 1993. LIR Regensburg.

Marx, J. (1994a): Die Auswertung des Benutzertests zur natürlichsprachlichen Komponente (inkl Vorschlagsmodus). WING-IIR Arbeitsbericht Nr. 49, März 1994. LIR Regensburg.

Marx, J. (1994b): Die Empirische Methode in WING-IIR. WING-IIR Arbeitsbericht Nr. 23 (Überarbeitung in Arbeit). LIR Regensburg.

Marx, J. (1995): Bidirektionale Sprache. Faktenrecherche und Informationsdarstellung durch dynamische Erzeugung korrigierbarer Zustandsanzeigen in natürlicher Sprache. Dissertation. Universität Regensburg.

Maybury, M. (ed.) (1993): Intelligent Multimedia Interfaces. AAAI Press/MIT Menlo Park, CA.

Meyer, B. (1990): Objektorientierte Softwareentwicklung. Wien et al.

Mittermaier, E. (1995): Planbasierte intelligente Hilfe. Design und empirische Fundierung auf der Basis eines symbiotischen Gesamtsystems. Schriften zur Informationswissenschaft, Bd. 10. Universitätsverlag Konstanz.

Owen, D. (1986): Naive Theories of Computation. In: Norman/Draper. S. 187-200.

Quibeldey-Cirkel, K. (1994): Das Objekt-Paradigma in der Informatik. Teubner, Stuttgart.

Paap, K. R., Roske-Hofstrand, R.J. (1988): Design of Menus. In: Helander, M. (ed.): Handbook of Human-Computer Interaction. Amsterdam et al. S. 205-235.

Pinker, S. (1990): A Theory of Graph Comprehension. In: Freedle, R. (ed.): Artificial Intelligence and the Future of Testing. Hillsdale, N.J. S: 73-126.

Roppel, S. (1996): Visualisierung und Adaption: Techniken zur Verbesserung der Interaktion mit hierarchisch strukturierter Information. Dissertation. Universität Regensburg.

Roppel, S., Wolff, C. (1992): Der erste multimodale Prototyp WING-M1. WING-IIR Arbeitsbericht Nr. 24, Juli 1992. LIR Regensburg.

Rosson, M.B., Alpert, S.R. (1990): The Cognitive Consequences of Object-Oriented Design. In: Human-Computer Interaction, Vol.5. S. 345-379.

Sager, W. (1991): Objektorientierte Programmierung. In: Computer Magazin 7, S. 34-40.

Siemens Nixdorf Informationssysteme (1992): Styleguide. Richtlinien zur Gestaltung von Benutzeroberflächen. Benutzerhandbuch. München.

Schudnagis, M. (1993): Vorüberlegungen zur Übertragung des WING-M2-Prototyps auf die HTM-Datenbank. WING-IIR Arbeitsbericht Nr. 47, Dezember 1993. LIR Regensburg.

Treu, S. (1992): Interface Structures: conceptual logical, and physical patterns applicabel to human-computer-interation. In: International Journal of man machine studies 37, S. 565-593.

Weingärtner, M. (1992): Objektorientierte, grafisch-direktmanipulative Textverarbeitung am Beispiel N/JOY. Regensburg (masch.).

Whiteside, J. et al. (1988): Usability Engineering: Our Experience and Evolution. In: Helander, M. (ed.): Handbook of Human-Computer Interaction. Amsterdam et al., S. 791-816.

Whorf, B.L. (1956): Language, Thought and Reality: Selected Papers (edited by J.B. Carroll). New York.

Winblad, A.L. et al. (1990): Object-Oriented Software. Addison-Wesley, Reading, MA.

Wolff, Chr. (1996): Graphisches Faktenretrieval mit Liniendiagrammen. Gestaltung und Evaluierung eines experimentellen Rechercheverfahrens auf der Grundlage kognitiver Theorien der Graphenwahrnehmung. Schriften zur Informationswissenschaft, Bd. 24. Universitätsverlag Konstanz.

Zloof, M.M. (1983): The Query by Example Concept for User-Oriented Business Systems. In: Coombs, M.J., Sime, M.E. (eds.): Designing for Human-Computer Communication. London, S. 285-309.